



计 算 机 类 本 科 规 划 教 材

# 计算方法

---

## (第2版)

---

◆ 李桂成 编著



电子工业出版社

PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

<http://www.phei.com.cn>

计算机类本科规划教材

# 计 算 方 法

## (第2版)

李桂成 编著

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

## 内 容 简 介

本书比较全面地介绍了现代科学与工程计算中常用的数值计算方法。全书共分 12 章，主要内容有：引论、计算方法的数学基础、方程求根、解线性方程组的直接法、解线性方程组的迭代法、函数插值、函数逼近、数值积分与数值微分、常微分方程初值问题的数值解法、矩阵特征值计算、函数优化计算和 MATLAB 编程基础及其在计算方法中的应用。

本书知识体系完整，从简要回顾与计算方法有关的数学基础知识，到介绍现代计算软件 MATLAB 在本领域中的应用，书中每个算法都配有结构化流程图，大部分算法给出了 MATLAB 语言和 C 语言的源代码，书后附有上机实验题目。可从华信教育资源网（[www.hxedu.com.cn](http://www.hxedu.com.cn)）免费下载的教学资源包括：电子教案、各章习题解答和模拟试题。

本书可作为高等院校理工科计算机、电子信息类及近电类本科和研究生教材使用，也可供从事科学与工程计算的科技工作者和研究人员参考。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

## 图书在版编目（CIP）数据

计算方法 / 李桂成编著. —2 版. —北京：电子工业出版社，2013.8

计算机类本科规划教材

ISBN 978-7-121-20328-2

I. ①计… II. ①李… III. ①计算方法—高等学校—教材 IV. ①0241

中国版本图书馆 CIP 数据核字（2013）第 094733 号

责任编辑：冉 哲

印 刷：

装 订：

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本：787×1 092 1/16 印张：20.25 字数：518.4 千字

印 次：2013 年 8 月第 1 次印刷

印 数：3 000 册 定价：41.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 [zltz@phei.com.cn](mailto:zltz@phei.com.cn)，盗版侵权举报请发邮件至 [dbqq@phei.com.cn](mailto:dbqq@phei.com.cn)。

服务热线：（010）88258888。

# 前言

随着科学技术的飞速发展和计算机的广泛应用,现代科学已呈现出理论科学、实验科学和计算科学三足鼎立的局面。作为计算科学的重要手段和工具,计算方法已成为当代理工科大学生必备的基础和技能。从20世纪80年代起,“计算方法”就成为信息和计算机等专业本科生的专业基础课。作者从事“计算方法”教学20余载,在教学过程中发现,学生在学习“计算方法”课程时,存在三个明显的问题:一是学生以前学过的数学知识大多已淡忘,翻开《高等数学》和《线性代数》课本,也不知从何看起;二是在实践环节中,学生用以前学过的程序设计语言编写计算方法实验程序,感到非常吃力;三是许多学生对计算方法中的算法,只知其然,不知其所以然。鉴于此,作者将自己一直使用的讲义进行整理,编写了这本教材,以尝试解决这些问题。

本书自2005年出版以来,深受读者喜爱和同行专家关注,作者也在本科和研究生教学中使用本书。经过多年的教学实践,收到了比较满意的效果,总体反映良好,但也发现一些有待改进之处。为了更好地适应教学和自学的需要,作者认真听取了关心本书的同行专家和读者的建议,决定修订再版。

本书第2版的内容主要有以下变动:

(1) 为了适应不同专业和层次的教学需求,增加了函数逼近、矩阵特征值计算和函数优化计算三章内容,在教学中可根据实际情况灵活调整。

(2) 删除了解线性方程组的平方根法和超松弛迭代法、函数插值的低阶含导数项的插值、解常微分方程的线性多步法等内容。

(3) 本着通俗易懂、利于教学、方便使用的原则,对部分内容进行了改写或重新编排。

(4) 对附录A中的计算方法实验进行了补充、细化和完善。

(5) 增加或改写了一些算法、程序、测试例题和部分习题。

全书共分12章:

第1章引言介绍计算方法与经典数学的差异,以及误差理论基础,主要包括误差的度量、分类、传播和数值计算的若干原则;

第2章简明、系统地介绍计算方法的数学基础,主要包括微积分、微分方程和线性代数的有关概念和结论;

第3章介绍非线性方程的数值解法,主要包括二分法、迭代法和牛顿法;

第4章介绍解线性方程组的直接法,主要包括消去法、矩阵分解法和误差估计;

第5章介绍解线性方程组的迭代法,主要包括雅可比迭代法、高斯-塞德尔迭代法,以及迭代公式的收敛性;

第6章介绍插值函数,主要包括拉格朗日插值、牛顿插值、埃尔米特插值和样条插值;

第7章介绍函数逼近,主要包括最佳一致逼近、最佳平方逼近及离散数据的曲线拟合;

第8章介绍数值积分和数值微分,主要包括牛顿-柯特斯求积公式、高斯求积公式、龙贝格求积公式及插值型求导公式;

第9章介绍常微分方程初值问题的数值解法,主要包括欧拉方法及其变形、龙格-库塔方法及单步法的收敛性和稳定性;



第 10 章介绍矩阵特征值计算, 主要包括幂法、QR 方法及雅克比方法;

第 11 章介绍函数优化计算, 主要包括一元函数优化计算、多元函数优化计算;

第 12 章介绍 MATLAB 编程基础及其在计算方法中的应用, 主要包括 MATLAB 编程环境、矩阵计算、图形功能, 以及用 MATLAB 实现计算方法中的基本算法, 并对这些算法配有 MATLAB 和 C 语言源代码。

讲授全书内容, 需要 40~60 学时, 另外, 实验需要 12~18 学时。

本书的特点是:

(1) 知识结构完整。既有计算方法所需的数学知识, 又有现代数值计算软件 MATLAB 在计算方法中的应用, 一书在手, 无须其他资料。

(2) 注重实践和应用。书中提供约 40 个 MATLAB 和 6 个 C 语言源代码实例, 以实现本书介绍的主要算法, 并对结果进行比较, 测试数据全部选自本书的例题; 附录 A 中有精心设计的 11 个实验, 供读者选用。

(3) 方便教学。全书每章都有内容提要、教学建议和本章小结, 说明每章的重点、难点、选学内容和所需教学时数。书后附录 A 中的 11 个实验, 可以加强计算方法课程的实践环节。本书还配有电子教案、模拟试题和各章习题解答, 供教师使用, 可以登录华信教育资源网([www.hxedu.com.cn](http://www.hxedu.com.cn))注册后免费下载。

(4) 便于自学。本书注重介绍算法的来龙去脉、基本思路和推导过程, 每个算法都配有结构化流程图, 并在第 12 章中有示例源代码和测试例题供读者参考。

特别值得一提的是, 书中对比了 C 语言和 MATLAB 实现各算法的优劣, 通过这种对比, 读者有望深刻理解“计算方法”的精髓。

本书可作为大学本科计算机、电子信息和近电类专业的教材使用, 也可供从事科学与工程计算的科技工作者和研究人员参考。

在本书第 2 版出版之际, 作者特别感谢梁吉业教授对本书的长期关注和大力支持, 感谢他对本书提出的宝贵意见和许多合理化的建议。电子工业出版社的高级编辑冉哲对本书的出版做了大量工作, 在此表示感谢。

由于作者水平有限, 书中难免有错误和疏漏之处, 恳请读者指正。

作 者

# 目 录

## 第 1 章 引论 .....1

### 1.1 从数学到计算 .....1

### 1.2 误差理论初步 .....5

#### 1.2.1 误差的来源 .....5

#### 1.2.2 误差的度量 .....6

#### 1.2.3 误差的传播 .....9

#### 1.2.4 数值稳定性 .....11

### 1.3 数值计算的若干原则 .....11

#### 1.3.1 避免两个相近数相减 .....12

#### 1.3.2 避免用绝对值过小的数作为 除数 .....12

#### 1.3.3 要防止大数“吃掉”小数 .....13

#### 1.3.4 简化计算步骤, 提高计算效率 .....14

#### 1.3.5 使用数值稳定的算法 .....14

### 本章小结 .....16

### 习题 1 .....16

## 第 2 章 计算方法的数学基础 .....18

### 2.1 微积分的有关概念和定理 .....18

#### 2.1.1 数列与函数的极限 .....18

#### 2.1.2 连续函数的性质 .....20

#### 2.1.3 罗尔定理和微分中值定理 .....20

#### 2.1.4 积分加权平均值定理 .....21

### 2.2 微分方程的有关概念和定理 .....22

#### 2.2.1 基本概念 .....22

#### 2.2.2 初值问题解的存在唯一性 .....23

### 2.3 线性代数的有关概念和定理 .....23

#### 2.3.1 线性相关和线性无关 .....23

#### 2.3.2 方阵及其初等变换 .....25

#### 2.3.3 线性方程组解的存在唯一性 .....27

#### 2.3.4 特殊矩阵 .....29

#### 2.3.5 方阵的逆及其运算性质 .....30

#### 2.3.6 矩阵的特征值及其运算性质 .....31

#### 2.3.7 对称正定矩阵 .....34

#### 2.3.8 对角占优矩阵 .....35

#### 2.3.9 向量和连续函数的内积 .....36

#### 2.3.10 向量、矩阵和连续函数的范数 .....37

#### 2.3.11 向量序列与矩阵序列的极限 .....42

### 本章小结 .....43

### 习题 2 .....43

## 第 3 章 方程求根 .....45

### 3.1 引言 .....45

### 3.2 二分法 .....46

### 3.3 迭代法 .....50

#### 3.3.1 不动点迭代 .....50

#### 3.3.2 迭代法的收敛性 .....51

#### 3.3.3 迭代法的改善 .....57

### 3.4 牛顿迭代法 .....59

#### 3.4.1 牛顿迭代公式及其几何意义 .....59

#### 3.4.2 牛顿迭代公式的收敛性 .....60

#### 3.4.3 重根情形 .....63

### 3.5 弦截法 .....65

### 本章小结 .....66

### 习题 3 .....66

## 第 4 章 解线性方程组的直接法 .....68

### 4.1 引言 .....68

### 4.2 高斯消去法 .....69

#### 4.2.1 顺序高斯消去法 .....69

#### 4.2.2 主元素高斯消去法 .....73

#### 4.2.3 高斯-约当消去法 .....75

### 4.3 矩阵三角分解法 .....77

#### 4.3.1 高斯消去法与矩阵三角分解 .....77

#### 4.3.2 直接三角分解法 .....78

### 4.4 解三对角方程组的追赶法 .....82

### 4.5 误差分析 .....85

#### 4.5.1 病态方程组与条件数 .....85

#### 4.5.2 病态方程组的解法 .....89

本章小结	90
习题 4	90
<b>第 5 章 解线性方程组的迭代法</b>	92
5.1 引言	92
5.2 雅可比迭代法	94
5.3 高斯-塞德尔迭代法	95
5.4 迭代法的收敛性	97
本章小结	104
习题 5	104
<b>第 6 章 函数插值</b>	107
6.1 引言	107
6.1.1 插值问题	107
6.1.2 插值多项式的存在唯一性	108
6.2 拉格朗日插值	109
6.2.1 线性插值与抛物插值	109
6.2.2 拉格朗日插值	111
6.2.3 插值余项与误差估计	113
6.3 牛顿插值	117
6.4 埃尔米特插值	121
6.5 分段低次插值	123
6.5.1 高次插值与龙格现象	123
6.5.2 分段线性插值	124
6.5.3 分段三次埃尔米特插值	126
6.6 样条函数插值	128
6.6.1 三次样条插值函数	128
6.6.2 三次样条插值函数的求法	130
本章小结	133
习题 6	133
<b>第 7 章 函数逼近</b>	137
7.1 引言	137
7.2 函数的内积与正交多项式	138
7.2.1 权函数和函数的内积	138
7.2.2 正交函数系	138
7.2.3 勒让德多项式	140
7.2.4 切比雪夫多项式	141
7.3 最佳一致逼近	142

7.3.1 基本概念	142
7.3.2 线性最佳一致逼近多项式	143
7.3.3 近似最佳一致逼近多项式	145
7.4 最佳平方逼近	146
7.4.1 基本概念	146
7.4.2 最佳平方逼近函数	147
7.5 离散数据的曲线拟合	149
7.5.1 曲线拟合问题	149
7.5.2 多项式拟合	150
7.5.3 正交多项式拟合	152
本章小结	153
习题 7	154
<b>第 8 章 数值积分与数值微分</b>	155
8.1 引言	155
8.1.1 数值求积的必要性	155
8.1.2 数值积分的基本思想	156
8.1.3 代数精度	156
8.1.4 插值型求积公式	158
8.2 牛顿-柯特斯求积公式	160
8.2.1 牛顿-柯特斯公式的导出	160
8.2.2 牛顿-柯特斯公式的误差估计	162
8.3 复合求积公式	164
8.3.1 复合梯形求积公式	165
8.3.2 复合辛普生求积公式	166
8.4 外推算法与龙贝格算法	168
8.4.1 变步长的求积公式	168
8.4.2 外推算法	169
8.4.3 龙贝格求积公式	170
8.5 高斯求积公式	174
8.5.1 高斯点与高斯求积公式	174
8.5.2 高斯-勒让德求积公式	175
8.5.3 高斯求积公式的稳定性和收敛性	178
8.6 数值微分	179
8.6.1 中点公式	179
8.6.2 插值型微分公式	181
本章小结	183
习题 8	183

## 第9章 常微分方程初值问题的

### 数值解法.....187

#### 9.1 引言.....187

#### 9.2 欧拉公式.....189

##### 9.2.1 欧拉公式及其意义.....189

##### 9.2.2 欧拉公式的变形.....190

#### 9.3 单步法的局部截断误差和方法的阶.....193

#### 9.4 龙格-库塔方法.....196

##### 9.4.1 龙格-库塔方法的基本思想.....196

##### 9.4.2 二阶龙格-库塔方法的推导.....196

##### 9.4.3 四阶经典龙格-库塔方法.....199

#### 9.5 单步法的收敛性和稳定性.....201

##### 9.5.1 单步法的收敛性.....202

##### 9.5.2 单步法的稳定性.....204

#### 本章小结.....207

#### 习题9.....207

## 第10章 矩阵特征值计算.....210

#### 10.1 引言.....210

#### 10.2 幂法及反幂法.....212

##### 10.2.1 幂法.....212

##### 10.2.2 反幂法.....215

#### 10.3 QR方法.....216

##### 10.3.1 反射变换.....217

##### 10.3.2 矩阵的QR分解.....218

##### 10.3.3 QR方法.....220

#### 10.4 雅可比方法.....221

##### 10.4.1 平面旋转矩阵.....221

##### 10.4.2 雅可比方法及其改进.....223

#### 本章小结.....225

#### 习题10.....226

## 第11章 函数优化计算.....227

#### 11.1 引言.....227

#### 11.2 一元函数优化计算.....228

##### 11.2.1 牛顿法.....228

##### 11.2.2 拟牛顿法.....230

##### 11.2.3 黄金分割法.....231

#### 11.3 多元函数优化计算.....232

##### 11.3.1 多元函数有最优解的条件.....232

##### 11.3.2 多元函数数值求解的原则.....233

##### 11.3.3 梯度法.....234

##### 11.3.4 牛顿法.....236

##### 11.3.5 共轭方向法.....238

##### 11.3.6 拟牛顿法(变尺度法).....240

#### 本章小结.....242

#### 习题11.....243

## 第12章 MATLAB编程基础及其在

### 计算方法中的应用.....244

#### 12.1 MATLAB简介.....244

#### 12.2 命令窗口和基本命令.....245

#### 12.3 变量、常量和数据类型.....246

#### 12.4 数值运算.....247

##### 12.4.1 向量运算.....247

##### 12.4.2 矩阵运算.....248

#### 12.5 符号运算.....251

##### 12.5.1 字符串运算.....251

##### 12.5.2 符号表达式运算.....252

##### 12.5.3 符号矩阵运算.....255

##### 12.5.4 符号微积分运算.....256

##### 12.5.5 方程求解.....258

#### 12.6 图形可视化.....260

##### 12.6.1 二维图形绘制.....260

##### 12.6.2 三维图形绘制.....261

#### 12.7 程序设计.....262

##### 12.7.1 命令文件与函数文件.....262

##### 12.7.2 控制语句.....263

##### 12.7.3 调试方法.....265

#### 12.8 MATLAB在计算方法中的应用.....266

##### 12.8.1 方程求根.....266

##### 12.8.2 解线性方程组的直接法.....270

##### 12.8.3 解线性方程组的迭代法.....275

##### 12.8.4 函数插值.....278

##### 12.8.5 函数逼近.....281

##### 12.8.6 数值积分.....283

##### 12.8.7 常微分方程的数值解法.....287

##### 12.8.8 矩阵特征值问题计算.....291



12.8.9 函数优化计算	297
本章小结	299
习题 12	300
<b>附录 A 计算方法实验</b>	<b>301</b>
实验 1 方程求根	302
实验 2 解方程组的直接法	303
实验 3 解线性方程组的迭代法	304
实验 4 插值问题	305

实验 5 曲线拟合	306
实验 6 数值积分	307
实验 7 数值微分	308
实验 8 求解常微分方程的初值问题	309
实验 9 求解三对角线性方程组	310
实验 10 矩阵特征值问题计算	312
实验 11 函数优化计算	313
<b>参考文献</b>	<b>315</b>

# 第1章 引 论



## 学习要点

(1) 数值计算的特点。计算方法研究的对象是用计算机求解各类数学问题, 它既具有纯数学的抽象性与严密性特点, 又具有应用的广泛性与实验的技术性特点。

(2) 误差理论。误差的来源、误差的度量、误差的传播。

(3) 数值计算的若干原则。避免两相近数相减和绝对值太小的除数, 简化计算步骤, 使用数值稳定的算法。



## 教学建议

本章是“计算方法”课程的开始, 要求学生了解计算方法的特点和研究对象, 重点学习误差的基本概念和性质, 掌握绝对误差、相对误差和有效数字的关系, 了解数值计算的基本原则。建议学时数为 2~4 学时。

## 1.1 从数学到计算

随着计算机的广泛应用和科学技术的高速发展, 大量复杂的科学计算问题呈现在人们面前, 要完成这些工作, 仅靠人的自身努力是不可能的, 必须借助于计算机这一人类有史以来最伟大的科技发明, 而使计算机有效解决科学计算问题的关键技术是数值计算方法。让我们首先从下面的几个例子谈起。

**引例 1 (例 1.1.1)** 设  $f(x)$  在  $[a, b]$  区间单调连续, 并且  $f(a) \cdot f(b) < 0$ , 求方程  $f(x) = 0$  在区间  $[a, b]$  内的根。

**解** 由微积分的知识可知, 方程  $f(x) = 0$  在  $[a, b]$  区间内有唯一的实根, 解此问题与函数  $f(x)$  的形式有关。

若  $f(x) = ax^2 + bx + c$ , 则由一元二次方程的求根公式知, 方程  $f(x) = 0$  的根为

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

若  $f(x) = x^3 - 1 = (x-1)(x^2 + x + 1)$ , 由代数知识可知,  $f(x) = 0$  的根为

$$x_1 = 1, \quad x_2 = \frac{-1 + i\sqrt{3}}{2} = \omega, \quad x_3 = \frac{-1 - i\sqrt{3}}{2} = \omega^2 \quad (i^2 = -1) \quad (1.1.1)$$

若  $f(x) = x^3 + px + q$ , 可令  $x = \alpha + \beta$ , 代入得

$$(\alpha + \beta)^3 + p(\alpha + \beta) + q = 0$$

展开得

$$\alpha^3 + 3\alpha^2\beta + 3\alpha\beta^2 + \beta^3 + p\alpha + p\beta + q = 0$$

因式分解得  $(\alpha^3 + \beta^3 + q) + (\alpha + \beta)(3\alpha\beta + p) = 0$ ，因为  $x = \alpha + \beta$  不能为 0，可令

$$\alpha^3 + \beta^3 + q = 0, \quad 3\alpha\beta + p = 0$$

则  $\alpha\beta = -\frac{p}{3}$ ，记为  $\alpha^3\beta^3 = -\frac{p^3}{27}$ ，以及  $\alpha^3 + \beta^3 = -q$ 。

由韦达定理，可将  $\alpha^3, \beta^3$  看成是二次方程  $z^2 + qz - \frac{p^3}{27} = 0$  的根，得

$$\begin{aligned}\alpha^3 &= \frac{-q}{2} + \sqrt{\frac{q^2}{4} + \frac{q^3}{27}}, & \beta^3 &= \frac{-q}{2} - \sqrt{\frac{q^2}{4} + \frac{q^3}{27}} \\ \alpha &= \sqrt[3]{\frac{-q}{2} + \sqrt{\frac{q^2}{4} + \frac{q^3}{27}}}, & \beta &= \sqrt[3]{\frac{-q}{2} - \sqrt{\frac{q^2}{4} + \frac{q^3}{27}}}\end{aligned}$$

$$\text{由此可得} \quad x_1 = \alpha + \beta = \sqrt[3]{\frac{-q}{2} + \sqrt{\frac{q^2}{4} + \frac{q^3}{27}}} + \sqrt[3]{\frac{-q}{2} - \sqrt{\frac{q^2}{4} + \frac{q^3}{27}}} \quad (1.1.2)$$

另外两个共轭复根为： $x_2 = \alpha\omega + \beta\omega^2$ ， $x_3 = \alpha\omega^2 + \beta\omega$ ，其中  $\omega$  为式 (1.1.1) 的值。

例如  $f(x) = x^3 - x - 1$ ，则由式 (1.1.2)，其实根为

$$x = \sqrt[3]{\frac{1}{2} + \sqrt{\frac{23}{108}}} + \sqrt[3]{\frac{1}{2} - \sqrt{\frac{23}{108}}}$$

若  $f(x) = ax^3 + bx^2 + cx + d$ ，则可将方程  $f(x) = 0$  两边除以  $a$ ，并设  $x = y - \frac{b}{3a}$ ，可将原方程

化为如下形式

$$y^3 + py + q = 0$$

由此解出  $y_1, y_2, y_3$  后得  $x_i = y_i - \frac{b}{3a} (i=1, 2, 3)$ 。

若  $f(x)$  为四次多项式，可想而知，其求根公式会更加复杂，不过仍然可以求解。遗憾的是，挪威年轻的数学家阿贝尔 (Abel) 在 1826 年证明，若  $f(x)$  为五次及以上的多项式，则代数方程  $f(x) = 0$  的根，不能由系数的初等函数表示，即五次及以上的代数方程无求根公式。

但是，在历史上，冥王星的发现归结为求解一个八次方程。不仅如此，人们还要求超越方程，例如确定轨道上行星位置的开普勒 (Kepler) 方程为

$$x = q \sin x + a \quad (\text{其中 } 0 < q < 1, \text{ 为椭圆轨道的偏心率})$$

实际上，用本书第 3 章介绍的牛顿法，可以求解此类方程。

**引例 1 说明，某些数学问题在理论上没有解决的方法，解决此类问题必须用数值计算方法求其近似值。**

**引例 2 (例 1.1.2) 求解  $n$  阶线性方程组  $Ax = b$ 。其中**

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ a_{31} & a_{32} & \cdots & a_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, \quad X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix}$$

由线性代数的知识，根据克莱姆 (Cramer) 法则，若系数矩阵  $A$  是非奇异矩阵，即  $\det(A) \neq 0$ ，则线性方程组  $Ax = b$  有唯一的解。

$$x_1 = \frac{\det(A_1)}{\det(A)}, x_2 = \frac{\det(A_2)}{\det(A)}, \dots, x_n = \frac{\det(A_n)}{\det(A)} \quad (1.1.3)$$

其中矩阵

$$A_j = \begin{pmatrix} a_{11} & \cdots & a_{1(j-1)} & b_1 & a_{1(j+1)} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2(j-1)} & b_2 & a_{2(j+1)} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & \cdots & a_{n(j-1)} & b_n & a_{n(j+1)} & \cdots & a_{nn} \end{pmatrix} \quad (j=1,2,\dots,n)$$

是用常向量  $b$  代换系数矩阵  $A$  的第  $j$  列后所得的  $n$  阶方阵。式中,  $a_{1(j-1)}$  表示第 1 行第  $j-1$  列, 其余类同。

这就意味着要解决此问题, 需要计算  $\det(A), \det(A_1), \det(A_2), \dots, \det(A_n)$ , 共  $n+1$  个行列式, 而每个行列式展开后有  $n!$  项, 每项有  $n$  个数, 需要  $n-1$  次乘法。如果忽略加、减、除法的次数, 仅计算乘法的次数, 则为  $(n+1) \times n! \times (n-1)$ 。例如, 当  $n=20$  时, 乘法次数为  $21 \times 20! \times 19 \approx 9.7 \times 10^{20}$ 。

若用每秒完成 12.5 万次乘除法的计算机计算, 则需  $9.7 \times 10^{20} \div (1.25 \times 10^5) \approx 7.8 \times 10^{15}$  秒。而 1 年  $= 365 \times 24 \times 3600 \approx 3.2 \times 10^7$  秒, 若计算时间用年计算, 则需要  $7.8 \times 10^{15} \div (3.2 \times 10^7) \approx 2.4 \times 10^8$  年, 约为两亿四千万年。

如果用每秒 1 亿次乘除法的银河 I 型巨型计算机计算此问题需 30 万年, 用每秒 10 亿次乘除法的银河 II 型巨型计算机计算此问题需 3 万年, 即使用目前世界上最快的“天河 2 号”计算机进行计算, 也需要数月。而现代科学技术, 如大型喷气客机、石油储藏模拟、流体涡度计算等, 未知数常常超过百万。显然, 仅仅只靠提高计算机的计算速度不是解决此类问题的主要手段。

实际上, 用本书第 4 章介绍的高斯消去法, 同样用每秒 12.5 万次乘除法的计算机求解 20 阶线性方程组只需要大约 0.02 秒的时间。

**引例 2 说明, 某些数学问题在理论上有解决的方法, 但在实际中并不实用, 必须寻找新的行之有效的计算方法。**

**引例 3 (例 1.1.3)** 求定积分  $I_n = \int_0^1 \frac{x^n}{x+5} dx \quad (n=1,2,\dots,20)$ 。

**解** 因为

$$\begin{aligned} I_n + 5I_{n-1} &= \int_0^1 \frac{x^n}{x+5} dx + 5 \int_0^1 \frac{x^{n-1}}{x+5} dx = \int_0^1 \frac{x^n + 5x^{n-1}}{x+5} dx = \int_0^1 \frac{x^{n-1}(x+5)}{x+5} dx \\ &= \int_0^1 x^{n-1} dx = \frac{1}{n} x^n \Big|_0^1 = \frac{1}{n} \end{aligned}$$

由此可得递推公式: 
$$I_n = \frac{1}{n} - 5I_{n-1} \quad (1.1.4)$$

并且 
$$I_0 = \int_0^1 \frac{1}{x+5} dx = \ln(x+5) \Big|_0^1 = \ln 6 - \ln 5 = \ln \frac{6}{5} \approx 0.182322$$

由微积分的知识, 我们知道  $I_n$  有如下性质:

①  $I_n > 0$  (因为被积函数  $f(x)$  在  $(0, 1)$  上非负)

②  $I_n$  单调递减 (当  $n_1 > n_2$  时,  $I_{n_1} > I_{n_2}$ )

③  $\lim_{n \rightarrow \infty} I_n = 0$  (因为  $\lim_{n \rightarrow \infty} \frac{x^n}{x+5} = 0, x \in [0, 1]$ )

④  $\frac{1}{6n} < I_{n-1} < \frac{1}{5n} \quad (n > 1)$



由式 (1.1.4) 得:  $I_{n-1} = \frac{1}{5n} - \frac{1}{5}I_n < \frac{1}{5n}$ , 又因为  $I_n$  单调递减  $I_n = \frac{1}{n} - 5I_{n-1} < I_{n-1}$ , 故  $I_{n-1} > \frac{1}{6n}$ 。

现在用两种方法计算  $I_n$ 。

方法 A: 按式 (1.1.4) 直接从  $I_1$  计算到  $I_{20}$ , 其计算结果见表 1.1.1。

表 1.1.1 计算结果 1

$n$	$I_n$	$n$	$I_n$	$n$	$I_n$	$n$	$I_n$
1	0.088 392 2	6	0.024 323 9	11	0.017 324 7	16	-10.1569
2	0.058 038 9	7	0.021 237 8	12	-0.003 290 22	17	50.8433
3	0.043 138 7	8	0.018 810 9	13	-0.093 374 2	18	-254.161
4	0.034 306 3	9	0.017 056 6	14	-0.395 442	19	1270.86
5	0.028 468 6	10	0.014 716 9	15	2.043 88	20	-6354.23

方法 B: 由  $I_n$  的性质④知:  $I_{n-1} \approx \left(\frac{1}{5n} + \frac{1}{6n}\right)/2$

所以 
$$I_{20} \approx \frac{\frac{1}{6 \times 21} + \frac{1}{5 \times 21}}{2} = 0.00873016$$

然后利用递推公式 
$$I_{n-1} = \frac{1}{5n} - \frac{1}{5}I_n \quad (1.1.5)$$

自  $I_{20}$  计算到  $I_1$ , 其计算结果见表 1.1.2。

表 1.1.2 计算结果 2

$n$	$I_n$	$n$	$I_n$	$n$	$I_n$	$n$	$I_n$
19	0.008 253 97	14	0.011 229 2	9	0.016 926 5	4	0.034 306 3
18	0.008 875 52	13	0.012 039 9	8	0.018 836 9	3	0.043 138 7
17	0.009 336 01	12	0.012 976 6	7	0.021 232 6	2	0.058 038 9
16	0.009 897 50	11	0.014 071 3	6	0.024 325 0	1	0.088 392 2
15	0.010 520 5	10	0.015 367 6	5	0.028 468 4	0	0.182 322

现在我们对这两种方法进行比较, 方法 A 的初值  $I_0$  具有 6 位有效数字, 比较精确, 但由该方法产生的数值解自  $I_{12}$  开始出现负值, 且其绝对值逐渐增加, 这显然与  $I_n$  的性质相矛盾, 因此由方法 A 计算的结果不符合原问题的要求。而方法 B 的初始值  $I_{20}$  取的是一个近似的平均值, 误差比较大, 但所得结果不但完全符合原问题的要求, 而且最后  $I_0$  的值与方法 A 的初始值相同, 具有较高的精度。其根本的原因是方法 A 由式 (1.1.4) 直接计算, 从  $I_{n-1}$  到  $I_n$  每向前推进一步, 若  $I_{n-1}$  有误差, 则其计算值的舍入误差增长 5 倍, 误差的放大传播导致最终的结果与原问题的真值相悖, 因此, 是不稳定的。而方法 B 由式 (1.1.5) 从  $I_n$  到  $I_{n-1}$  计算, 每向后推进一步, 若  $I_n$  有误差的话, 其舍入误差便减少为原来的 1/5, 因此, 获得了与原问题的性质一致的数值结果, 是稳定的。

实际上, 用本书第 1 章的后续知识, 可以避免方法 A 现象的发生。

**引例 3 说明, 某些数学问题即使在实践中有解决的方法, 仍然需要进行误差分析。**

从以上 3 个引例可以看出, 在解决科学计算问题时, 经典的数学方法受到了极大的限制, 虽然, 近代数学家们提出了许多理论与方法, 证明了一些问题的解存在、唯一以及解的某些特

征,但仍然只是对解的性质给出了某些定性的描述,而科技工作者和工程师却需要真实的、定量的数据。因此,许多数学问题的解决必须借助于计算机,而且要选择合理、有效的方法。我国著名的计算科学家石钟慈院士指出:“计算不仅仅只是作为验证理论模型的正确手段,大量的实例表明它已是重大科学发现的一种重要手段”,“科学计算与实验,理论三足鼎立,相辅相成,成为当今科学发现的三大方法”。

本书介绍的计算方法就是专门研究各种数学问题的计算机解法(数值解法),包括方法的构造和求解过程的理论分析及软件实现,它是计算数学的一个主要部分,包括方法的收敛性、稳定性以及误差分析等。计算方法既具有纯数学的抽象性与严密性的特点,又具有应用的广泛性与实验的技术性特点,因此,在学习计算方法时,要充分考虑计算机的特点,使所构造的算法,应该只包括计算机能直接处理的算术运算和逻辑运算,并严格控制计算的复杂性,最后还要在相关数学理论的基础上对误差进行分析。

由于数学的学科十分广泛,所出现的数学问题也各不相同。本书只涉及工程和科学实验中常见的数学问题,其中包括线性方程组、函数插值、微积分、微分方程、非线性方程、矩阵的特征值、函数逼近以及优化计算等,这些问题是解决其他数学问题的基础。

由于本书的内容包括了微积分、微分方程、非线性方程和线性方程组的数值计算方法,因此,在介绍这些常见的数值计算方法之前,第2章简要介绍了微积分、常微分方程和线性代数方面的基本内容,以便读者查阅。

## 1.2 误差理论初步

### 1.2.1 误差的来源

用数值计算方法求解数学问题,不可避免地会产生误差。实际上,在各种实际问题的求解过程中,误差的产生是绝对的,精确值却是相对的,产生误差的原因一般有以下几种。

#### 1. 模型误差

例如求一个鸡蛋的表面积。首先要建一个数学模型,可以近似用球的表面积公式计算。但鸡蛋与球的形状差别较大,为了减少误差,可以用椭球的表面积公式计算。但鸡蛋的形状是一头大,一头小,仍然与椭球的形状不同,为了进一步减少误差,可以将鸡蛋的曲线画出来,利用曲面积分公式计算,但仍然会产生误差。这种**数学模型的解与实际问题的解之间出现的误差,称为模型误差**。

#### 2. 测量误差

在建立数学模型以后,接下来就要进行一些数据的测量。例如,为了求鸡蛋的表面积,就要进行测量相关数据,若用球的表面积公式计算,则要测量其半径;若用椭球表面积计算,则要测量长半轴和短半轴;若要用曲面积分公式计算,则要测量积分区间等。由于测量手段的限制,因此在实际测量中,总会产生误差。这种**在测量具体数据时产生的误差称为测量误差**。

#### 3. 截断误差(也称方法误差)

当用数学模型不能求出问题的精确解时,就要用数值计算方法求解,例如用梯形法求定积分

$$I = \int_a^b f(x) dx$$

$$I \approx I_1 = \frac{b-a}{2} [f(a) + f(b)] \quad (1.2.1)$$

实际上, 这是用过两点  $(a, f(a)), (b, f(b))$  的直线  $l(x)$  代替被积函数  $f(x)$  得到的积分值, 即

$$I_1 = \int_a^b l(x) dx$$

在本书第 8 章中, 我们会知道  $I$  与  $I_1$  之间的误差为

$$R_1 = I - I_1 = -\frac{(b-a)^3}{12} f''(\eta) \quad \eta \in (a, b)$$

$R_1$  称为梯形求积公式的方法误差。这种**数学模型的准确解与数值计算方法的准确解之间的误差称为截断误差**。因为截断误差是方法固有的, 所以又称为方法误差。对某数学问题的数值解进行误差估计, 主要是指方法误差, 这是本书要讨论的重点。

#### 4. 舍入误差

由于计算机字长的限制, 某些数 (如无理数  $\pi$ ) 不能在计算机内精确表示, 计算结果必然也会产生误差。例如, 梯形求积公式 (1.2.1) 在用计算机求解  $f(a), f(b)$  时, 一般只能得到它们的近似值  $\tilde{f}(a)$  和  $\tilde{f}(b)$ , 加上式 (1.2.1) 计算的误差, 最终只能得到  $I_1$  的近似值  $I_2$ 。

$$I_1 \approx I_2 = \frac{b-a}{2} [\tilde{f}(a) + \tilde{f}(b)]$$

$I_1$  与  $I_2$  之间的误差就是舍入误差。这种**由于计算机字长的限制而产生的误差, 称为舍入误差**。

针对不同的数值计算方法, 误差估计的侧重点也不同。例如, 在线性方程组的数值求解中, 主要讨论输入数据的误差和舍入误差的传播, 而在数值积分和微分中, 重点分析各种方法的截断误差。

### 1.2.2 误差的度量

对于同一个数学问题, 采用不同的方法会得出不同的结果。衡量某种方法优劣的标准之一, 是看其结果的误差是否较小。一般度量误差的标准有三种形式。

#### (1) 绝对误差与绝对误差限

**定义 1.2.1** 设  $x$  为某量的精确值,  $x^*$  是它的一个近似值, 则称  $E(x^*) = x - x^*$  为近似值  $x^*$  的绝对误差, 简称误差。由于精确值  $x$  是未知的, 因此  $x^*$  的绝对误差  $E(x^*)$  一般也是求不出来的。但是如果能求出  $x^*$  误差的一个范围  $E(x^*) = |x - x^*| \leq \delta(x^*)$ , 则称  $\delta(x^*)$  为近似值  $x^*$  的绝对误差限, 简称误差限。

例如, 设  $x = \pi = 3.1415926535 \dots$ , 若取  $x$  的一个近似值  $x^* = 3.14159$ , 则

$$\delta(x^*) = |x - x^*| \leq 0.5 \times 10^{-5}$$

称  $x^*$  的误差限为  $0.5 \times 10^{-5}$ 。

一个近似数的误差限并不唯一, 通常取满足  $|x - x^*| \leq \frac{1}{2} \times 10^n$  ( $n$  为整数) 的最小值。

#### (2) 相对误差与相对误差限

绝对误差有时不能完全刻画一个近似数的精确程度, 例如测量一个书桌和一个体育场的面积, 误差都是  $1\text{cm}^2$ , 显然, 后者的测量更精确, 因此, 决定某量的近似值的精度, 除了考虑绝

对误差的大小外,还要考虑该量自身的大小,为此引入相对误差的概念。

**定义 1.2.2** 设  $x$  为某量的精确值,  $x^*$  是它的一个近似值,则称  $E_r(x^*) = \frac{x-x^*}{x^*} (x \neq 0)$  为  $x^*$  的相对误差。

由于精确值  $x$  是未知的,因此,在实际计算中常取  $E_r(x^*) = \frac{x-x^*}{x^*}$  作为  $x^*$  的相对误差。若  $E_r(x^*)$  的绝对值小于某个已知正数  $\delta_r(x^*)$ , 即  $|E_r(x^*)| = \left| \frac{x-x^*}{x^*} \right| \leq \delta_r(x^*)$ , 则称  $\delta_r(x^*)$  为近似值  $x^*$  的相对误差限。

例如, 设  $x = \pi = 3.14159265358\cdots$ , 取  $x$  一个近似值  $x^* = 3.14159$ , 则

$$\delta_r(x^*) = \left| \frac{x-x^*}{x^*} \right| \leq 0.8 \times 10^{-6}$$

称  $x^*$  的相对误差限为  $0.8 \times 10^{-6}$ 。

### (3) 有效数字

当某量的精确值的位数较多时,我们通常采用“四舍五入”的方法取  $x$  的前面若干位,作为  $x$  的近似值。

例如,  $x = 3.14159265358\cdots$

取 1 位  $x_1 = 3$   $\delta(x_1) \approx 0.14 \leq 0.5$

取 5 位  $x_5 = 3.1416$   $\delta(x_5) \approx 0.000007 < 0.00005$

取 10 位  $x_{10} = 3.141592654$   $\delta(x_{10}) \approx 0.00000000042 < 0.0000000005$

这些近似值的误差限都不超过该近似值的最后 1 位数字的半个单位,则称它们都是有效数字,由此得有效数字的定义。

**定义 1.2.3** 如果近似值  $x^*$  的误差限不超过某位的半个单位,若该位数字到  $x^*$  的第一位非零数字共有  $n$  位,那么这  $n$  位数字称为  $x^*$  的有效数字,并称  $x^*$  具有  $n$  位有效数字。

在计算机中参加运算的数往往要进行规格化表示,因此,有效数字也可以定义如下。

**定义 1.2.4** 设  $x^*$  是  $x$  的一个近似值,写成规格化形式

$$x^* = \pm 10^k \times 0.a_1 a_2 \cdots a_n \cdots \quad (1.2.2)$$

式中  $a_i (i=1,2,\cdots)$  是  $0 \sim 9$  之间的整数,且  $a_1 \neq 0$ ,  $k$  为整数。

如果

$$|x-x^*| \leq \frac{1}{2} \times 10^{k-n} \quad (1.2.3)$$

则称  $x^*$  为  $x$  的具有  $n$  位有效数字的近似值。

**例 1.2.1** 设  $x = \sqrt{200} = 14.142\cdots$ ,  $x^* = 14.1$

$y = \lg 2 = 0.30102\cdots$ ,  $y^* = 0.3010$

$z = e^{-5} = 0.0067379\cdots$ ,  $z^* = 0.00673$

求各近似值的有效数字。

**解** 方法 1:

由于  $\delta(x^*) = |x-x^*| = 0.042 \leq 0.05$ , 小于十分位的半个单位,因此, 14.1 每位都是有效数字,故  $x^*$  有 3 位有效数字。

由于  $\delta(y^*) = |y-y^*| = 0.00002 < 0.00005$ , 小于万分位的半个单位,因此, 0.3010 中小数点后



均为有效数字, 故  $y^*$  有 4 位有效数字。

由于  $\delta(z^*) = |z - z^*| = 0.0000079 < 0.00005$ , 小于万分位的半个单位, 因此, 0.00673 中的 6 和 7 为有效数字, 而 3 不是有效数字, 故  $z^*$  有 2 位有效数字。

方法 2:

因为  $x^* = 0.141 \times 10^2$ ,  $k = 2$ , 而  $|x - x^*| \leq \frac{1}{2} \times 10^{-1}$ ,  $k - n = -1$ , 所以  $n = 3$ , 故  $x^*$  有 3 位有效数字。

因为  $y^* = 0.3010 \times 10^0$ ,  $k = 0$ , 而  $|y - y^*| \leq \frac{1}{2} \times 10^{-4}$ ,  $k - n = -4$ , 所以  $n = 4$ , 故  $y^*$  有 4 位有效数字。

因为  $z^* = 0.673 \times 10^{-2}$ ,  $k = -2$ , 而  $|z - z^*| \leq \frac{1}{2} \times 10^{-4}$ ,  $k - n = -4$ , 所以  $n = 2$ , 故  $z^*$  有 2 位有效数字。

从上面的例子可以看出, 有效数字的个数与小数点的位置及小数点后的位数无关。不过, 从式 (1.2.3) 知, 在  $k$  相同的情况下,  $n$  越大, 则  $k - n$  越小, 所以有效数字越多, 绝对误差就越小。

下面讨论误差的三种度量之间的关系。由于绝对误差限、相对误差限和有效数字都是用来度量近似数的误差的, 因此它们之间必然存在着一定的联系。实际上, 由相对误差的定义  $E_r(x^*) = \frac{E(x^*)}{x^*}$  可知, 相对误差限与绝对误差限的关系是:  $\delta_r(x^*) = \delta(x^*)/x^*$ 。由有效数字的定义可知, 有效数字与绝对误差的关系是: 若近似值  $x^* = \pm 10^k \times 0.a_1a_2 \cdots a_n \cdots$  的绝对误差限为  $\delta(x^*) = |x - x^*| \leq \frac{1}{2} \times 10^{k-n}$ , 则  $x^*$  具有  $n$  位有效数字, 而有效数字与相对误差限的关系可由以下的定理得到。

**定理 1.2.1** 设  $x$  的近似值为式 (1.2.2) 的规格化形式,

$$\textcircled{1} \text{ 若 } x^* \text{ 有 } n \text{ 位有效数字, 则 } \frac{|x - x^*|}{|x^*|} \leq \frac{1}{2a_1} \times 10^{1-n}. \quad (1.2.4)$$

$$\textcircled{2} \text{ 若 } \frac{|x - x^*|}{|x^*|} \leq \frac{1}{2(a_1 + 1)} \times 10^{1-n}, \text{ 则 } x^* \text{ 至少具有 } n \text{ 位有效数字}. \quad (1.2.5)$$

**证明** 由  $x^*$  的规格化形式可得

$$a_1 \times 10^{k-1} \leq |x^*| \leq (a_1 + 1) \times 10^{k-1} \quad (1.2.6)$$

所以当  $x^*$  有  $n$  位有效数字时, 有

$$\frac{|x - x^*|}{|x^*|} \leq \frac{0.5 \times 10^{k-n}}{a_1 \times 10^{k-1}} = \frac{1}{2a_1} \times 10^{1-n}$$

结论①得证。

又由式 (1.2.5) 和式 (1.2.6) 知

$$|x - x^*| \leq \frac{1}{2(a_1 + 1)} \times 10^{1-n} \times |x^*| \leq \frac{1}{2(a_1 + 1)} \times 10^{1-n} \times (a_1 + 1) \times 10^{k-1} = 0.5 \times 10^{k-n}$$

根据有效数字的定义,  $x^*$  具有  $n$  位有效数字。

结论②得证。

**例 1.2.2** 要使  $1/19$  的近似值的相对误差限不超过 0.1%, 应取几位有效数字?

解 因为  $1/19 = 0.05263157\cdots$ ，由于  $a_1 = 5$ ，要使  $\delta_r(x^*) \leq 0.001$ ，则

$$\delta_r(x^*) \leq \frac{10^{-(n-1)}}{2a_1} = \frac{1}{10} \times 10^{-n+1} = 10^{-n} \leq 0.001$$

由上式可得  $n \geq 3$ 。

所以，只要对  $1/19$  取近似值  $0.0526$ ，即取 3 位有效数字，其相对误差限就小于  $0.1\%$ 。

### 1.2.3 误差的传播

在近似数的运算过程中，初始数据（或已知数据）的误差对计算结果有着直接的影响，这就是所谓的误差传播问题，误差的传播是否可以控制，标志着一种数值计算方法的优劣。

#### 1. 函数的误差

设  $x^*$  是精确值  $x$  的一个近似值， $y^*$  是  $y$  的一个近似值，现在分别对一元函数  $f(x)$  和二元函数  $f(x, y)$  的误差进行分析。

设函数  $f(x)$  在  $x^*$  的邻域上连续可微，由一阶泰勒展开式的近似式

$$f(x) \approx f(x^*) + f'(x^*)(x - x^*)$$

得

$$|f(x) - f(x^*)| \leq |f'(x^*)| \cdot |x - x^*|$$

由此得  $f(x)$  的近似函数值  $f(x^*)$  的误差限和相对误差限分别有如下的估计式

$$\begin{cases} \delta f(x^*) \leq |f'(x^*)| \cdot \delta(x^*) \\ \delta_r f(x^*) \leq \left| \frac{\delta f(x^*)}{f(x^*)} \right| = \left| \frac{f'(x^*)}{f(x^*)} \right| \cdot \delta(x^*) \end{cases} \quad (1.2.7)$$

式中， $\delta(x^*)$  为  $x^*$  的误差限。

设函数  $f(x, y)$  在  $(x^*, y^*)$  的邻域上连续可微，则由二元函数的一阶泰勒展开式的近似式

$$f(x, y) \approx f(x^*, y^*) + \frac{\partial f(x^*, y^*)}{\partial x}(x - x^*) + \frac{\partial f(x^*, y^*)}{\partial y}(y - y^*)$$

得

$$|f(x, y) - f(x^*, y^*)| \leq \left| \frac{\partial f(x^*, y^*)}{\partial x} \right| \cdot |x - x^*| + \left| \frac{\partial f(x^*, y^*)}{\partial y} \right| \cdot |y - y^*|$$

由此得  $f(x, y)$  的近似函数值  $f(x^*, y^*)$  的误差限和相对误差限分别有如下的估计式

$$\begin{cases} \delta(f(x^*, y^*)) \leq \left| \frac{\partial f(x^*, y^*)}{\partial x} \right| \cdot \delta(x^*) + \left| \frac{\partial f(x^*, y^*)}{\partial y} \right| \cdot \delta(y^*) \\ \delta_r |f(x^*, y^*)| \leq \frac{\delta(f(x^*, y^*))}{|f(x^*, y^*)|} \end{cases} \quad (1.2.8)$$

式中， $\delta(x^*), \delta(y^*)$  分别为  $x^*, y^*$  的误差限。

例 1.2.3 设  $x^* > 0$ ， $x^*$  的相对误差限为  $\varepsilon_r$ ，求  $\ln x^*$  的误差限。

解 设  $f(x) = \ln x$ ，由式 (1.2.7) 得

$$\delta(f(x^*)) = |f(x) - f(x^*)| \leq |f'(x^*)| \cdot \delta(x^*) = \frac{1}{|x^*|} |x - x^*| = \varepsilon_r,$$

所以  $\ln x^*$  的误差限为  $\varepsilon_r$ 。

例 1.2.4 计算  $f = (\sqrt{2} - 1)^6$ , 取  $\sqrt{2} \approx 1.4$ , 直接计算  $f$  和利用以下式子  $\frac{1}{(\sqrt{2} + 1)^6}$ ,  $(3 - 2\sqrt{2})^3$ ,  $\frac{1}{(3 + 2\sqrt{2})^3}$ ,  $99 - 70\sqrt{2}$  计算, 哪个误差最小?

解 所给 5 个式子分别看做

$$f(x) = (x - 1)^6, f_1(x) = (x + 1)^{-6}, f_2(x) = (3 - 2x)^3, f_3(x) = (3 + 2x)^{-3}, f_4(x) = 99 - 70x$$

取  $x = \sqrt{2}$  的近似值  $x^* = 1.4$ ,  $|x - x^*| \leq 0.02 = \delta$ , 利用误差估计式 (1.2.7) 得

$$\delta f(x^*) \approx |f'(x^*)(x - x^*)| \leq 6(x^* - 1)^5 \delta \leq 0.062\delta$$

同理

$$\delta f_1(x^*) \leq 6(x^* - 1)^{-7} \delta \leq 0.014\delta$$

$$\delta f_2(x^*) \leq 6(3 - 2x^*)^2 \delta \leq 0.24\delta$$

$$\delta f_3(x^*) \leq 6(3 + 2x^*)^{-4} \delta \leq 0.00531\delta$$

$$\delta f_4(x^*) \leq 70\delta$$

由此可见, 用  $\frac{1}{(3 + 2\sqrt{2})^3}$  计算时误差最小。

## 2. 算术运算的误差

用计算机进行数值计算时, 由于所有的函数计算都必须转化成算术运算, 因此算术运算的误差估计是最基本的。

分别设

$$f(x, y) = x \pm y, f(x, y) = x \cdot y, f(x, y) = x / y$$

$x^*$  为  $x$  的近似值,  $y^*$  为  $y$  的近似值, 则由式 (1.2.8) 不难得出加、减、乘、除运算的误差限和相对误差限的估计式

$$\begin{cases} \delta(x^* \pm y^*) \leq \delta(x^*) + \delta(y^*) \\ \delta_r(x^* \pm y^*) \leq \frac{\delta(x^*) + \delta(y^*)}{|x^* \pm y^*|} \end{cases} \quad (1.2.9)$$

和

$$\begin{cases} \delta(x^* y^*) \leq |y^*| \delta(x^*) + |x^*| \delta(y^*) \\ \delta_r(x^* y^*) \leq \frac{\delta(x^*)}{|x^*|} + \frac{\delta(y^*)}{|y^*|} = \delta_r(x^*) + \delta_r(y^*) \end{cases} \quad (1.2.10)$$

$$\begin{cases} \delta\left(\frac{x^*}{y^*}\right) \leq \frac{1}{|y^*|} \delta(x^*) + \left|\frac{x^*}{y^{*2}}\right| \delta(y^*) \\ \delta_r\left(\frac{x^*}{y^*}\right) \leq \frac{\delta(x^*)}{|x^*|} + \frac{\delta(y^*)}{|y^*|} = \delta_r(x^*) + \delta_r(y^*) \end{cases} \quad (1.2.11)$$

上述公式总结为: 和、差的误差限不超过各误差限的和, 积、商的相对误差限不超过各相对误差限的和。

**例 1.2.5** 经过四舍五入得出  $x_1 = 6.1025, x_2 = 80.115$ 。试问：

(1) 它们各具有几位有效数字？(2) 求  $x_1 + x_2, x_1 - x_2, x_1 x_2, \frac{x_1}{x_2}$  的绝对误差限。

**解** 记  $x_1$  和  $x_2$  对应的精确值分别是  $x_1^*$  和  $x_2^*$ ，则有  $|x_1^* - x_1| \leq \frac{1}{2} \times 10^{-4}$  和  $|x_2^* - x_2| \leq \frac{1}{2} \times 10^{-3}$ ，故  $x_1$  和  $x_2$  各具有 5 位有效数字。

再根据误差限的估计式得：
$$\delta(x_1 \pm x_2) \leq \delta x_1 + \delta x_2 \leq \frac{1}{2} \times 10^{-4} + \frac{1}{2} \times 10^{-3} = 0.00055$$

$$\delta(x_1 x_2) \leq |x_2| \delta x_1 + |x_1| \delta x_2 \leq 80.115 \times \frac{1}{2} \times 10^{-4} + 6.1025 \times \frac{1}{2} \times 10^{-3} = 0.007057$$

$$\delta\left(\frac{x_1}{x_2}\right) \leq \frac{|x_2| \delta x_1 + |x_1| \delta x_2}{|x_2|^2} \leq \frac{0.007057}{80.115^2} = 0.10995 \times 10^{-5}$$

## 1.2.4 数值稳定性

误差的传播能否得到控制，是误差分析的重要内容，也是衡量一个算法优劣的重要指标，可以用算法的数值稳定性表示误差传播的控制。

**定义 1.2.5** 对于某个数值算法，如果输入数据的误差，在计算过程中不断扩大而难以得到控制，则称该算法是数值不稳定的，否则是数值稳定的。如果某算法在一定的条件下才是数值稳定的，则称该算法是条件稳定（相对稳定）的。若在任何条件下，某算法都是数值稳定的，则称该算法是无条件稳定（绝对稳定）的。

例如在 1.1 节的引例 3 中，方法 A 是数值不稳定的，而方法 B 是数值稳定的。

**例 1.2.6** 序列  $\{x_n\}$  满足递推关系  $x_{n+1} = 8x_n + 6$  ( $n = 0, 1, 2, \dots$ )。若  $x_0 = \sqrt{2} \approx 1.41$ ，计算到  $x_{10}$  时，误差有多大？此计算公式数值稳定吗？

**解** 设  $x_0 = \sqrt{2}$  的近似值  $x_0^* = 1.41$

$$\delta(x_0^*) = |x_0 - x_0^*| \leq \frac{1}{2} \times 10^{-2} = \varepsilon$$

$$\delta(x_1^*) = |x_1 - x_1^*| = |(8x_0 + 6) - (8x_0^* + 6)| = 8|x_0 - x_0^*| \leq 8\varepsilon$$

$$\delta(x_{10}^*) = |x_{10} - x_{10}^*| \leq 8|x_9 - x_9^*| \leq \dots \leq 8^{10}|x_0 - x_0^*| \leq 8^{10}\varepsilon$$

由于此递推公式每计算一步，误差增长 8 倍，第 10 步时增长到  $8^{10}$  倍，因此，该算法是数值不稳定的。

## 1.3 数值计算的若干原则

由误差的基本知识我们知道，数值计算的每步运算都可能产生误差，而每个数学问题的解决，往往需要经过成千上万次的运算。我们不可能，也没有必要对每步的误差进行误差分析。但是通过长期对误差产生的原因以及对误差传播规律的分析，人们总结出了在数值计算中若干原则，这些原则有助于鉴别数值计算结果的可靠性和稳定性，也有助于防止误差蔓延现象的发生。这些原则对数值计算问题的求解具有指导意义。



### 1.3.1 避免两个相近数相减

由算术运算的误差公式

$$\delta_r(x^* - y^*) \leq \frac{\delta(x^* - y^*)}{|x^* - y^*|}$$

可知, 如果  $x^*$  和  $y^*$  的值非常接近, 则  $x^* - y^*$  的相对误差限会增大, 从而引起有效数字的严重丢失。

例 1.3.1 当  $x=1000$ , 取 4 位有效数字时计算

$$y = \sqrt{x+1} - \sqrt{x} \quad (1.3.1)$$

解 由于  $\sqrt{x+1} \approx 31.64$ ,  $\sqrt{x} \approx 31.62$ , 二者相减得  $y=0.02$ , 此结果的有效数字只有 1 位, 丢失了 3 位有效数字, 可以想象计算结果  $y$  的绝对误差和相对误差将变得非常大, 这将影响计算结果的精度。

如果将式 (1.3.1) 改写为

$$y = \sqrt{x+1} - \sqrt{x} = \frac{1}{\sqrt{x+1} + \sqrt{x}}$$

按此公式计算, 可得  $y=0.01581$ , 此时计算结果  $y$  具有 4 位有效数字。由此可见, 适当改变原有的计算公式, 可以避免两相近数相减, 进而避免有效数字的丢失, 确保结果的精度。

类似地, 当  $|x|$  很小时, 可以用下式计算

$$1 - \cos x = 2 \sin^2 \left( \frac{x}{2} \right)$$

或

$$1 - \cos x = \frac{x^2}{2!} - \frac{x^4}{4!} + \frac{x^6}{6!} - \frac{x^8}{8!} + \cdots$$

当  $x_1, x_2$  很接近时, 可以用下式计算

$$\ln x_1 - \ln x_2 = \ln \frac{x_1}{x_2}$$

当  $|x|$  很大, 而  $\varepsilon$  很小时, 可以用下式计算

$$\sin(x + \varepsilon) - \sin x = 2 \cos\left(x + \frac{\varepsilon}{2}\right) \sin \frac{\varepsilon}{2}$$

一般地, 当  $f(x) \approx f(x^*)$  时, 可以用泰勒展开式计算

$$f(x) - f(x^*) = f'(x^*)(x - x^*) + \frac{f''(x^*)}{2!}(x - x^*)^2 + \cdots$$

### 1.3.2 避免用绝对值过小的数作为除数

设

$$z = \frac{x^*}{y^*}$$

由商的误差估计式 (1.2.11) 知

$$\delta\left(\frac{x^*}{y^*}\right) \leq \frac{1}{|y^*|} \delta(x^*) + \frac{|x^*|}{|y^{*2}|} \delta(y^*)$$

所以在进行除法计算时, 如果除数太小, 即使除数的误差很小, 则商的误差也可能放大很大。

### 例 1.3.2 计算

$$\frac{x^*}{y^*} = \frac{2.7182}{0.001}$$

解

$$\frac{x^*}{y^*} = \frac{2.7182}{0.001} = 2718.2$$

若分母变为 0.0011, 即分母的变化只有 0.0001, 则

$$\frac{x^*}{y^*} = \frac{2.7182}{0.0011} = 2471.1$$

可见, 商却引起了巨大的变化。因此, 在算法设计时, 要尽量避免在算法的计算公式中用绝对值过小的数作为除数。

### 1.3.3 要防止大数“吃掉”小数

由数在计算机内的表示等知识可知, 计算机在进行算术计算时, 首先要把参加运算的数对阶, 即把两数都写成绝对值小于 1, 而阶码相同的数。

例如,  $x=10^9+1$  必须改写成  $x=0.1\times 10^{10}+0.0000000001\times 10^{10}$ 。如果计算机只能表示 8 位小数, 则只能算出  $x=0.1\times 10^{10}$ , 大数吃掉了小数。这种情形要尽量避免。

**例 1.3.3** 求一元二次方程  $x^2-(10^9+1)x+10^9=0$  的根。

**解** 利用因式分解将原方程改写为

$$(x-1)(x-10^9)=0$$

得此方程的根

$$x_1=10^9, x_2=1$$

但若利用求根公式

$$x_{1,2} = \frac{10^9+1 \pm \sqrt{(10^9+1)^2-4\times 10^9}}{2}$$

用只能表示 8 位小数的计算机计算, 由于对阶有

$$10^9+1 \approx 10^9, \sqrt{(10^9+1)^2-4\times 10^9} \approx 10^9$$

从而

$$x_1=10^9, x_2=0$$

显然结果是错误的。要防止大数“吃掉”小数, 应将计算公式

$$x_2 = \frac{10^9+1 - \sqrt{(10^9+1)^2-4\times 10^9}}{2}$$

改为

$$x_2 = \frac{2\times 10^9}{10^9+1 + \sqrt{(10^9+1)^2-4\times 10^9}}$$

则有

$$x_2 \approx \frac{2\times 10^9}{10^9+10^9} = 1$$

从而可以得到正确的结果。

### 1.3.4 简化计算步骤, 提高计算效率

简化计算步骤, 可以减少算法的计算量和误差的累积, 在 1.1 节的引例 2 中, 用克莱姆法则求解线性方程组, 其绝大多数乘法运算都是多余的, 是可以去掉的。

**例 1.3.4** 计算  $x^{255}$  的值。

**解** 若直接计算  $x^{255}$  的值, 则需要进行 254 次乘法。若采用公式

$$x^{255} = x \cdot x^2 \cdot x^4 \cdot x^8 \cdot x^{16} \cdot x^{32} \cdot x^{64} \cdot x^{128}$$

计算  $x^{255}$  的值, 除  $x$  外, 每个因数计算 1 次乘法, 共 7 次, 连乘计算 7 次乘法, 则总共只需做 14 次的乘法运算。

**例 1.3.5** 计算多项式的值。

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = \sum_{k=0}^n a_k x^k \quad (1.3.2)$$

**解** 若直接用上面的公式逐项求和运算。计算第  $k$  项  $a_k x^k$ , 需要  $k$  次乘法, 因此总共需要做  $\frac{1}{2}n(n+1)$  次乘法和  $n$  次加法。

但若将式 (1.3.2) 改写成如下形式

$$p_n(x) = x(x \cdots (x(a_n x + a_{n-1}) + a_{n-2}) + \cdots + a_1) + a_0$$

并设

$$\begin{cases} s_0 = a_n \\ s_k = s_{k-1} \cdot x + a_{n-k} \quad (k=1, 2, \cdots, n) \\ s_n = p(x_n) \end{cases}$$

则求  $s_k$  ( $k=1, 2, \cdots, n$ ) 只需 1 次乘法和 1 次加法, 从而求  $s_n = p(x_n)$  只需  $n$  次乘法和  $n$  次加法即可。上述算法由我国宋代科学家秦九韶首先提出。不过, 一些外文资料也称其为霍纳 (Hornor) 算法。

从上面两个例子可以看出简化公式的重要性。

**例 1.3.6** 计算例 1.2.4 中的  $f = (\sqrt{2} - 1)^6$ , 取  $\sqrt{2} \approx 1.4$ , 利用下列式子计算, 哪个得到的结果最好?

$$\frac{1}{(\sqrt{2}+1)^6}, (3-2\sqrt{2})^3, \frac{1}{(3+2\sqrt{2})^3}, 99-70\sqrt{2}$$

**解**  $99-70\sqrt{2}$ ,  $(3-2\sqrt{2})^3$  出现较相近的数相减, 故二者不可能得到最好的结论。  $\frac{1}{(\sqrt{2}+1)^6}$

与  $\frac{1}{(3+2\sqrt{2})^3}$  均不出现相近数相减, 但前者的乘法运算次数多, 而除法运算次数相同, 每次乘

除法运算必然引入新的舍入误差, 故  $\frac{1}{(3+2\sqrt{2})^3}$  将给出最好的运算结果。这一结论与例 1.2.4 的结论一致。

### 1.3.5 使用数值稳定的算法

在用计算机解决实际问题时, 运算次数成千上万。如果误差的传播得不到控制, 那么误差的累积会使问题的解答成为荒谬的, 尤其是某些病态问题 (如病态方程组), 舍入误差对其计算

结果往往有非常严重的影响。因此,在选择计算方案时,要特别谨慎。例如在 1.1 节的引例 3 中,方法 A 是数值不稳定的,而方法 B 是数值稳定的。

例 1.3.7 计算下列积分。

$$I_n = e^{-1} \int_0^1 x^n e^x dx \quad (n=0,1,2,\cdots,13)$$

解

$$I_n = \int_0^1 x^n e^{x-1} dx = \int_0^1 x^n d e^{x-1} = x^n e^{x-1} \Big|_0^1 - n \int_0^1 x^{n-1} e^{x-1} dx = 1 - n I_{n-1}$$

特别当  $n=0$  时,有

$$I_0 = \int_0^1 e^{x-1} dx = 1 - e^{-1} \approx 0.6321205$$

由微积分的知识不难得出积分  $I_n$  有下列特性:

①  $I_n > 0$

②  $I_n < \frac{1}{n+1} < I_{n-1}$

③ 当  $n \rightarrow \infty$  时,  $I_n \rightarrow 0$

现在用两种方法计算  $I_n$ 。

方法 A:

$$I_n = 1 - n I_{n-1}, \quad I_0 = 1 - e^{-1} = 0.6321205$$

初值的误差限为  $0.587 \times 10^{-7}$ , 计算结果见表 1.3.1 中的  $I_n(A)$  列。

方法 B:

$$I_{n-1} = \frac{1}{n}(1 - I_n)$$

$$\frac{1}{n+1} < I_{n-1} < \frac{1}{n}$$

取

$$I_{n-1} = \left( \frac{1}{n+1} + \frac{1}{n} \right) / 2 = \frac{2n+1}{2n(n+1)}$$

当  $n=14$  时,取初始值

$$I_{13} = \frac{2 \times 14 + 1}{2 \times 14 \times 15} = \frac{29}{420} \approx 0.0690476$$

其初值的误差限为  $-2.1 \times 10^{-3}$ , 计算结果见表 1.3.1 中的  $I_n(B)$  列。

表 1.3.1 计算结果

$n$	$I_n(A)$	$I_n(B)$	$n$	$I_n(A)$	$I_n(B)$
0	0.632 120 5	0.632 120 5	8	0.100 945 6	0.100 931 9
1	0.367 879 4	0.367 879 4	9	0.091 489 6	0.091 612 4
2	0.264 241 1	0.264 241 1	10	0.085 104	0.083 875 8
3	0.207 276 6	0.207 276 6	11	0.063 856	0.077 365 6
4	0.170 893 4	0.170 893 4	12	0.233 728	0.071 611 7
5	0.145 532 9	0.145 532 9	13	-2.038 464	0.069 047 6
6	0.126 802 6	0.126 802 3	14	29.538 496	
7	0.112 381 8	0.112 383 5			

对于方法 A, 若实际计算递推公式为

$$I_n^* = 1 - nI_{n-1}^*$$

式中,  $I_n^*$  为  $I_n$  的近似值, 则

$$I_n - I_n^* = -n(I_{n-1} - I_{n-1}^*) = \cdots = (-1)^n n!(I_0 - I_0^*)$$

可见, 初始数据的误差是按  $n!$  倍数增长的。

当  $n=13$  时,  $I_{13} - I_{13}^* = -13! \times (I_0 - I_0^*) = -6.227 \times 10^9 \times 0.587 \times 10^{-7} \approx 3.655 \times 10^2$

可见  $I_{13}^*$  的误差已把  $I_{13}$  的真值覆盖掉了, 所以方法 A 是数值不稳定的。

对于方法 B, 若实际计算时的递推公式为

$$I_{n-1}^* = \frac{1}{n}(1 - I_n^*)$$

式中,  $I_n^*$  为  $I_n$  的近似值, 则  $I_{n-1} - I_{n-1}^* = -\frac{1}{n}(I_n - I_n^*)$ , 所以

$$I_0 - I_0^* = -\frac{1}{1}(I_1 - I_1^*) = \cdots = (-1)^n \frac{1}{n!}(I_n - I_n^*)$$

可见, 初始数据的误差  $I_n^*$  传播给  $I_0^*$  是按  $\frac{1}{n!}$  的倍数而缩小的。

当  $n=0$  时,  $I_0 - I_0^* = -\frac{1}{13!} \times (I_{13} - I_{13}^*) = -1.6 \times 10^{-10} \times (-2.1 \times 10^{-3}) = 3.36 \times 10^{-13}$

即方法 B 算得的结果是可靠的, 它具有较好的数值稳定性。

## 本章小结

本章首先通过 3 个引例指出数学与计算的本质区别, 并说明数值计算方法是用计算机求解数学问题的主要技术, 在此基础上说明了计算方法的特点及研究对象。用数值计算方法求解数学问题的过程中, 必然产生误差, 因此本章介绍了误差的来源、误差的度量和误差传播的控制, 以及函数运算和算术运算对误差的影响等误差理论, 最后介绍了在进行数值运算中要注意的 5 个原则, 这些原则对数值计算过程具有指导意义。

## 习题 1

1.1 下列各数都是经过四舍五入得到的近似值, 试指出它们有几位有效数字, 并给出其误差限和相对误差限。

$$x_1^* = 1.1021, x_2^* = 0.031, x_3^* = 560.40$$

1.2 要使  $\sqrt{11}$  的近似值的相对误差限不超过 0.1%, 应取几位有效数字?

1.3 设原始数据的下列近似值每位都有有效数字:

$$a_1 = 1.1021, a_2 = 0.031, a_3 = 385.6, a_4 = 56.430$$

试计算: (1)  $a_1 + a_2 + a_4$  之和; (2)  $a_1 a_2 a_3$  之积, 并估计它们的相对误差限。

1.4 设  $x^*$  的相对误差为 2%, 求  $(x^*)^n$  的相对误差。

1.5 长方体的长、宽和高分别约为 50cm, 20cm 和 10cm, 测量误差满足什么条件时该长

方体表面积误差不超过  $1\text{cm}^2$ 。

1.6 正方形的边长大约为  $100\text{cm}$ ，应怎样测量才能使其面积误差不超过  $1\text{cm}^2$ ？

1.7 已测的某住房长为  $l^* = 4.32\text{m}$ ，宽为  $d^* = 3.12\text{m}$ ，已知  $|l - l^*| \leq 0.01\text{m}$ ， $|d - d^*| \leq 0.01\text{m}$ ，

试求住房面积  $S = ld$  的误差限和相对误差限。

1.8 求  $\sqrt{20}$  的近似有效数字，要求：(1) 使绝对误差限不超过  $0.01$ ；(2) 使相对误差限不超过  $0.01$ 。

1.9 已知  $|x|$  非常小于  $1$ ，下列计算  $y$  的公式哪个算得准？

(1) (A)  $y = \frac{1}{1+2x} - \frac{1-x}{1+x}$

(B)  $y = \frac{2x^2}{(1+2x)(1+x)}$

(2) (A)  $y = \frac{2|x|}{\sqrt{\frac{1}{|x|} + |x|} + \sqrt{\frac{1}{|x|} - |x|}}$

(B)  $y = \sqrt{\frac{1}{|x|} + |x|} - \sqrt{\frac{1}{|x|} - |x|}$

(C)  $y = \frac{2|x|^{\frac{3}{2}}}{\sqrt{1+x^2} + \sqrt{1-x^2}}$

(D)  $y = \frac{\sqrt{1+x^2} - \sqrt{1-x^2}}{\sqrt{|x|}}$

(3) (A)  $y = \frac{2\sin^2 x}{x}$

(B)  $y = \frac{1 - \cos 2x}{x}$

(4) (A)  $y = \ln \frac{1 - \sqrt{1-x^2}}{|x|}$

(B)  $y = \ln |x| - \ln(1 + \sqrt{1-x^2})$

(C)  $y = \ln \frac{|x|}{1 + \sqrt{1-x^2}}$

1.10 下列公式如何变形才能使数值计算得到比较精确的结果？

(1)  $x - \sin x$  ( $|x| \ll 1$ )

(2)  $\int_N^{N+1} \ln x dx = (N+1)\ln(N+1) - N\ln N - 1$  ( $N$  充分大)

1.11 设  $x_1 = 1.216$ ,  $x_2 = 3.654$ ，均具有  $3$  位有效数字，则  $x_1 x_2$  的相对误差限为\_\_\_\_\_。

1.12 分别用  $2.718\ 2811$  和  $2.718\ 282$  作为数  $e$  的近似值，则其有效位数分别有\_\_\_\_\_位和\_\_\_\_\_位；又取  $\sqrt{3} \approx 1.73$  ( $3$  位有效数字)，则  $|\sqrt{3} - 1.73| \leq$ \_\_\_\_\_。

1.13 已知近似值  $x_A = 2.4560$  是由真值  $x_T$  经舍入得到的，则相对误差限为\_\_\_\_\_。

1.14 为减少乘除法运算次数，应将算式  $y = 18 + \frac{3}{x-1} + \frac{5}{(x-1)^2} - \frac{7}{(x-1)^3}$  改写成\_\_\_\_\_；为减少舍入误差的影响，应将算式  $10 - \sqrt{99}$  改写成\_\_\_\_\_。

1.15 递推公式  $\begin{cases} y_0 = \sqrt{2} \\ y_n = 10y_{n-1} - 1 \end{cases} (n=1, 2, \dots)$ ，如果取  $y_0 = \sqrt{2} \approx 1.41$  进行计算，则计算到  $y_{10}$

时，误差限为\_\_\_\_\_；这个计算公式数值稳定还是不稳定？

## 第2章 计算方法的数学基础



### 学习要点

本章介绍了计算方法需要的数学基本知识, 主要包括:

(1) 微积分。数列与函数的极限、闭区间上连续函数的性质、微分中值定理、积分加权平均值定理。

(2) 微分方程。初值问题的概念、初值问题解的存在唯一性。

(3) 线性代数。线性空间及向量的相关性、矩阵的概念及相关的运算性质、向量的内积、向量和矩阵的范数。



### 教学建议

本章内容是学习后续章节的数学基础, 知识的组织精炼而系统。本章的教学重点是引领学生回忆、复习, 并结合所给例题巩固已学知识, 学时数由教师根据学生的知识水平灵活掌握, 建议学时数为 4~8 学时。

## 2.1 微积分的有关概念和定理

### 2.1.1 数列与函数的极限

**定义 2.1.1 (数列的极限)** 如果数列  $\{x_n\}$  与常数  $a$  有如下关系: 对于任意给定的正数  $\varepsilon$  (不论它多么小), 总存在正整数  $N$ , 使得对  $n > N$  时的一切  $x_n$ , 不等式

$$|x_n - a| < \varepsilon$$

都成立, 则称常数  $a$  是数列  $\{x_n\}$  的极限, 或者称数列  $\{x_n\}$  收敛于  $a$ , 记为

$$\lim_{n \rightarrow \infty} x_n = a \quad \text{或} \quad x_n \rightarrow 0 \quad (n \rightarrow \infty)$$

如果数列没有极限, 则称该数列是发散的。

**例 2.1.1** 设  $|q| < 1$ , 证明等比数列  $x_n = 1, q, q^2, \dots, q^{n-1}, \dots$  的极限是 0。

**证明** 任意给定  $\varepsilon > 0$  (设  $\varepsilon < 1$ ), 因为

$$|x_n - 0| = |q^{n-1} - 0| = |q|^{n-1}$$

要使  $|x_n - 0| < \varepsilon$ , 只要  $|q|^{n-1} < \varepsilon$ 。

上述不等式两边取自然对数, 得  $(n-1)\ln|q| < \ln\varepsilon$ 。因为  $|q| < 1$ ,  $\ln|q| < 0$ , 故  $n > 1 + \frac{\ln\varepsilon}{\ln|q|}$ ,

取  $N = \left\lceil 1 + \frac{\ln\varepsilon}{\ln|q|} \right\rceil$ , 则当  $n > N$  时, 就有

$$|\varepsilon^{n-1} - 0| < \varepsilon$$

即  $\lim_{n \rightarrow \infty} \varepsilon^{n-1} = 0$

收敛数列具有如下三个性质:

- ① 唯一性: 收敛数列  $\{x_n\}$  的极限唯一。
- ② 有界性: 收敛数列  $\{x_n\}$  一定有界。
- ③ 一致性: 收敛数列  $\{x_n\}$  与其任一子序列的极限相同。

一个数列是否有极限, 可以由以下三个定理来判断。

**定理 2.1.1 (夹逼定理)** 如果数列  $\{x_n\}, \{y_n\}, \{z_n\}$  满足下列条件:

$$(1) x_n \leq y_n \leq z_n \quad (n=1, 2, 3, \dots)$$

$$(2) \lim_{n \rightarrow \infty} x_n = a, \quad \lim_{n \rightarrow \infty} z_n = a$$

那么数列  $\{y_n\}$  的极限存在, 且  $\lim_{n \rightarrow \infty} y_n = a$ 。

**定理 2.1.2 (单调有界定理)** 单调有界数列必有极限。

**定理 2.1.3 (柯西定理)** 数列  $\{x_n\}$  收敛的充分必要条件是对于任意给定的正数  $\varepsilon$ , 存在着一个正整数  $N$ , 使得当  $n > N, m > N$  时, 有

$$|x_n - x_m| < \varepsilon$$

**例 2.1.2** 证明数列  $\{x_n\}$  的收敛性。

$$x_n = \frac{\cos 1}{1 \times 2} + \frac{\cos 2}{2 \times 3} + \dots + \frac{\cos n}{n \times (n+1)}$$

**证明** 设  $k$  为正整数, 对  $\forall \varepsilon > 0$ , 要求

$$|x_{n+k} - x_n| \leq \frac{1}{(n+1)(n+2)} + \frac{1}{(n+2)(n+3)} + \dots + \frac{1}{(n+k)(n+k+1)} = \frac{1}{n+1} - \frac{1}{n+k+1} < \frac{1}{n+1} < \varepsilon$$

取正整数  $N \geq \left\lceil \frac{1}{\varepsilon} - 1 \right\rceil$ , 当  $n > N$  时, 对任意的正整数  $k$ , 可使  $|x_{n+k} - x_n| < \varepsilon$ , 所以数列  $\{x_n\}$

的极限存在。

**定义 2.1.2 (函数的极限 1)** 设函数  $f(x)$  在点  $x_0$  的某一邻域内有定义, 如果对于任意给定的正数  $\varepsilon$  (不论它多么小), 总存在正数  $\delta$ , 使得对于适合不等式  $0 < |x - x_0| < \delta$  的一切  $x$ , 对应的函数值  $f(x)$  都满足不等式

$$|f(x) - A| < \varepsilon$$

那么常数  $A$  称为函数  $f(x)$  当  $x \rightarrow x_0$  时的极限, 记为

$$\lim_{x \rightarrow x_0} f(x) = A \quad \text{或} \quad f(x) \rightarrow A \quad (x \rightarrow x_0)$$

**例 2.1.3** 证明  $\lim_{x \rightarrow 1} \frac{x^2 - 1}{x - 1} = 2$ 。

**证明** 对于任意给定的正数  $\varepsilon$ , 不等式  $\left| \frac{x^2 - 1}{x - 1} - 2 \right| < \varepsilon$ , 约去非零点因子  $|x - 1|$  后, 就化为

$$|x + 1 - 2| = |x - 1| < \varepsilon, \quad \text{故取 } \delta = \varepsilon, \quad \text{则当 } 0 < |x - 1| < \delta \text{ 时, 有 } \left| \frac{x^2 - 1}{x - 1} - 2 \right| < \varepsilon, \quad \text{因此 } \lim_{x \rightarrow 1} \frac{x^2 - 1}{x - 1} = 2。$$

利用单调有界定理可以证明  $\lim_{x \rightarrow \infty} \left(1 + \frac{1}{x}\right)^x = e$ 。

**定义 2.1.3 (函数的极限 2)** 设函数  $f(x)$  当  $|x|$  大于某一正数时有定义, 如果对于任意给



定的正数  $\varepsilon$  (不论它多么小) 总存在着正数  $X$ , 使得对于适合不等式  $|x| > X$  的一切  $x$ , 对应的函数值  $f(x)$  都满足不等式

$$|f(x) - A| < \varepsilon$$

那么常数  $A$  称为函数  $f(x)$  当  $x \rightarrow \infty$  时的极限, 记做

$$\lim_{x \rightarrow \infty} f(x) = A \quad \text{或} \quad f(x) \rightarrow A \quad (x \rightarrow \infty)$$

**例 2.1.4** 证明  $\lim_{x \rightarrow \infty} \frac{1}{x} = 0$ 。

**证明** 设  $\varepsilon$  为任意给定的正数, 要证明存在正数  $X$ , 当  $|x| > X$  时, 不等式  $\left| \frac{1}{x} - 0 \right| < \varepsilon$  成立。

而此不等式等价于  $\left| \frac{1}{x} \right| < \varepsilon$ , 即  $|x| > \frac{1}{\varepsilon}$ , 因此取  $X = \frac{1}{\varepsilon}$ , 则对满足  $|x| > X = \frac{1}{\varepsilon}$  的一切  $x$ ,

不等式  $\left| \frac{1}{x} - 0 \right| < \varepsilon$  成立。所以  $\lim_{x \rightarrow \infty} \frac{1}{x} = 0$ 。

## 2.1.2 连续函数的性质

**定义 2.1.4 (函数连续)** 设函数  $y = f(x)$  在  $x_0$  的某一邻域内有定义。如果对于任意给定的正数  $\varepsilon$ , 总存在着正数  $\delta$ , 使得对于适合不等式  $|x - x_0| < \delta$  的一切  $x$ , 对应的函数值  $f(x)$  都满足不等式

$$|f(x) - f(x_0)| < \varepsilon$$

那么称函数  $f(x)$  在点  $x_0$  连续, 称  $x_0$  为函数的连续点。

如果  $f(x)$  在闭区间  $[a, b]$  内每个点连续, 则称  $f(x)$  在闭区间  $[a, b]$  内连续, 用  $C[a, b]$  表示在闭区间  $[a, b]$  内的所有连续函数的集合, 则  $f(x) \in C[a, b]$  表示  $f(x)$  在  $[a, b]$  内连续。

**定义 2.1.5 (一致连续性)** 设函数  $f(x)$  在区间  $I$  内有定义, 如果对于任意给定的正数  $\varepsilon$ , 总存在正数  $\delta$ , 使得对于在区间  $I$  内的任意两点  $x_1, x_2$ , 当  $|x_1 - x_2| < \delta$  时, 有

$$|f(x_1) - f(x_2)| < \varepsilon$$

则称函数  $f(x)$  在区间  $I$  内是一致连续的。

闭区间内的连续函数有如下性质:

(1) (极值定理) 在闭区间内的连续函数在该区间内一定有最大值和最小值。

(2) (有界性定理) 在闭区间内的连续函数在该区间内一定有界。

(3) (一致连续性定理) 在闭区间内的连续函数在该区间内也一致连续。

(4) (零点定理) 若函数  $f(x)$  在闭区间  $[a, b]$  内连续, 且  $f(a)$  与  $f(b)$  异号 (即  $f(a) \cdot f(b) < 0$ ),

则函数  $f(x)$  在开区间  $(a, b)$  内至少有一个零点, 即至少有一个数  $\xi$  ( $a < \xi < b$ ) 使  $f(\xi) = 0$ 。

零点定理也称为方程实根的存在性定理, 常用于判断一个方程的根的存在性。

显然, 如果  $f(x) \in [a, b]$  为单调函数, 且  $f(a) \cdot f(b) < 0$ , 则  $f(x)$  在  $[a, b]$  内的零点唯一。

## 2.1.3 罗尔定理和微分中值定理

**定理 2.1.4 (罗尔定理)** 如果函数  $f(x)$  在闭区间  $[a, b]$  内连续, 在开区间  $(a, b)$  内可导, 且在区间端点的函数值相等, 即  $f(a) = f(b)$ , 那么在  $(a, b)$  内至少有一点  $\xi$  ( $a < \xi < b$ ), 使得函数  $f(x)$  在该点的导数等于 0, 即  $f'(\xi) = 0$ 。

**定理 2.1.5 (拉格朗日微分中值定理)** 如果函数  $f(x)$  在闭区间  $[a, b]$  内连续, 在开区间  $(a, b)$  内可导, 那么在  $(a, b)$  内至少有一点  $\xi$  ( $a < \xi < b$ ) 使等式  $f(b) - f(a) = f'(\xi)(b - a)$  成立。

显然, 当  $f(a) = f(b)$ , 拉格朗日微分中值定理变成罗尔定理。

**定理 2.1.6 (泰勒中值定理)** 如果函数  $f(x)$  在含有  $x_0$  的某个开区间  $(a, b)$  内具有直到  $(n+1)$  阶的导数, 则当  $x$  在  $(a, b)$  内时,  $f(x)$  可以表示为  $(x - x_0)$  的一个  $n$  次多项式与一个余项  $R_n(x)$  之和

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n + R_n(x) \quad (2.1.1)$$

式中,  $R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{n+1}$  ( $\xi$  介于  $x_0$  与  $x$  之间,  $R_n(x)$  称为余项)。

式 (2.1.1) 称为泰勒公式。

特别地, 当  $x_0 = 0$  时, 泰勒公式变为较简单的形式, 即所谓的麦克劳林公式

$$f(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \cdots + \frac{f^{(n)}(0)}{n!}x^n + \frac{f^{(n+1)}(\varphi x)}{(n+1)!}x^{n+1} \quad (0 < \varphi < 1)$$

由泰勒中值定理, 可以求得 5 个常用的初等函数  $e^x, \sin x, \cos x, \ln(1+x)$  和  $(1+x)^a$  在  $x_0 = 0$  处的泰勒公式如下:

$$e^x = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} + \frac{e^\xi}{(n+1)!}x^{n+1} \quad (\xi \text{ 在 } 0 \text{ 和 } x \text{ 之间})$$

$$\sin x = x - \frac{x^3}{3!} + \cdots + (-1)^{n-1} \frac{x^{2n-1}}{(2n-1)!} + \frac{1}{(2n)!} \sin(\xi + \frac{2n\pi}{2})x^{2n} \quad (\xi \text{ 在 } 0 \text{ 和 } x \text{ 之间})$$

$$\cos x = 1 - \frac{x^2}{2!} + \cdots + (-1)^n \frac{x^{2n}}{(2n)!} + \frac{1}{(2n+1)!} \cos(\xi + \frac{(2n+1)\pi}{2})x^{2n+1} \quad (\xi \text{ 在 } 0 \text{ 和 } x \text{ 之间})$$

$$\ln(1+x) = x - \frac{x^2}{2} + \cdots + (-1)^{n-1} \frac{x^n}{n} + (-1)^n \frac{1}{(1+\xi)^{n+1}} \frac{x^{n+1}}{n+1} \quad (\xi \text{ 在 } 0 \text{ 和 } x \text{ 之间})$$

$$(1+x)^a = 1 + ax + \frac{a(a-1)}{2!}x^2 + \cdots + \frac{a(a-1)\cdots(a-n+1)}{n!}x^n + \frac{a(a-1)\cdots(a-n)}{(n+1)!}(1+\xi)^{a-n-1}x^{n+1}$$

( $\xi$  在 0 和  $x$  之间,  $a$  为任意实数)

### 2.1.4 积分加权平均值定理

**定理 2.1.7 (积分中值定理)** 如果函数  $f(x)$  在闭区间  $[a, b]$  内连续, 则在积分区间  $[a, b]$  内至少存在一个数  $\xi$ , 使下式成立

$$\int_a^b f(x)dx = f(\xi)(b-a) \quad (a \leq \xi \leq b)$$

**定理 2.1.8 (积分加权平均值定理)** 如果函数  $f(x)$  在闭区间  $[a, b]$  内连续, 积分权函数  $g(x)$  为在  $[a, b]$  内的可积函数, 且  $g(x)$  在  $[a, b]$  内不变号, 则在积分区间  $[a, b]$  内至少存在一点  $\xi$ , 使下式成立

$$\int_a^b f(x)g(x)dx = f(\xi) \int_a^b g(x)dx$$

显然, 当  $g(x)=1$  时, 积分加权平均值定理就变为积分中值定理。一般称  $f(\xi)$  为  $f(x)$  在  $[a, b]$  内的平均值。

## 2.2 微分方程的有关概念和定理

### 2.2.1 基本概念

方程有许多形式。例如方程  $ax^2+bx+c=0$  是代数方程, 求解此方程, 就是寻找满足此方程的未知量  $x$ 。再如从方程  $F(x, y)=0$  中确定  $y$  为  $x$  的函数, 解此方程, 不是寻找某个未知数, 而是寻找一个未知数的函数  $y(x)$ , 因此, 这类方程称为函数方程。还有一类方程, 方程中不仅含有自变量和未知函数, 还含有未知函数的导数 (或微分), 这类方程就称为微分方程。

例如, 放射性元素镭的衰变 (因镭不断放射出射线而逐渐减少其质量), 要求其质量的变化规律, 可设在某时刻  $t$ , 镭的质量为  $x(t)$ , 根据衰变规律, 衰变率  $-\frac{dx}{dt}$  与  $x(t)$  成正比, 故得

$$\frac{dx}{dt} = -kx \quad (k \text{ 为常数})$$

此方程中含有未知函数  $x(t)$  的导数, 因此它是微分方程。

又如, 以下方程都是微分方程

$$\frac{dy}{dx} = f(x, y) \quad (2.2.1)$$

$$\frac{d^2y}{dx^2} + p(x)\frac{dy}{dx} + q(x)y = f(x) \quad (2.2.2)$$

$$\frac{\partial^4 z}{\partial x^4} + 2\frac{\partial^2 z}{\partial x^2 \partial y^2} + \frac{\partial^4 z}{\partial y^4} = 0 \quad (2.2.3)$$

在微分方程中, 如果未知函数是一元函数, 则相应的微分方程称为常微分方程, 例如, 式 (2.2.1) 和式 (2.2.2) 就是常微分方程。如果未知函数是多元函数, 方程中就含有未知函数的偏导数, 则相应的微分方程就称为偏微分方程, 例如, 式 (2.2.3) 为偏微分方程。微分方程中所含未知函数的导数的最高次数, 称为微分方程的阶, 例如, 式 (2.2.1) 为一阶常微分方程, 式 (2.2.2) 为二阶常微分方程, 式 (2.2.3) 为四阶偏微分方程。本书只讨论一阶常微分方程的求解问题。

求微分方程的未知函数, 称为对微分方程求解。

**例 2.2.1** 求微分方程的解。

$$\frac{dy}{dx} = 2xy \quad (2.2.4)$$

**解** 将原方程改写为  $\frac{1}{y}dy = 2xdx$  两边积分得

$$\ln y(x) = x^2 + c_1$$

所以

$$y(x) = e^{x^2+c_1} = e^{c_1} \cdot e^{x^2} = ce^{x^2} \quad (2.2.5)$$

式中,  $c$  为任意常数。

由此例可以看出, 一个常微分方程可以有无穷多个解, 常常表示为含有任意常数的形式。一般地,  $n$  阶微分方程的解中可以含有  $n$  个任意常数, 这时的解称为微分方程的通解, 例如式

(2.2.5) 就是式 (2.2.4) 的通解。有时, 根据具体问题, 要求出微分方程的某一特定解, 称为特解, 这就必须确定通解中任意常数的值, 为此需要给出一定的条件, 称为定解条件。求某个微分方程满足一定的定解条件的特解, 这种问题称为定解问题。若定解条件是初值条件, 相应的定解问题就称为初值问题。若定解条件在自变量所在区间  $[a, b]$  的边界上, 则相应的定解问题就称为边值问题。

**定义 2.2.1** 形如

$$\begin{cases} \frac{dy}{dx} = f(x, y) \\ y(x_0) = y_0 \end{cases} \quad (2.2.6)$$

$$(2.2.7)$$

称为一阶常微分方程的初值问题。

解微分方程的初值问题, 就是在  $XY$  平面域  $D$  中, 对于给定的一点  $(x_0, y_0)$ , 求方程 (2.2.6) 的一个特解, 使它满足式 (2.2.7)。也可以称为, 求微分方程 (2.2.6) 过点  $(x_0, y_0)$  的一个积分曲线,  $(x_0, y_0)$  为初始值, 式 (2.2.7) 为初值条件。

## 2.2.2 初值问题解的存在唯一性

在求解常微分方程的初值问题之前, 首先应该了解此初值问题的解是否存在。因为对某些提得不恰当的初值问题, 解可能不存在, 或者解存在但并不唯一。例如, 对于如下问题

$$\begin{cases} \left(\frac{dy}{dx}\right)^2 + y = 0 \\ y(0) = 1 \end{cases} \quad (2.2.8)$$

$$(2.2.9)$$

其在  $x=0$  的任何一个邻域  $[-h, h]$  内, 都不可能实数解, 这是因为如果某个函数  $y(x)$  满足式 (2.2.9) 的初值, 则它在原点处不会满足微分方程 (2.2.8)。

再如对于如下初值问题

$$\begin{cases} \frac{dy}{dx} = y^{\frac{2}{3}} \\ y(0) = 0 \end{cases}$$

就同时有解  $y = \frac{x^3}{27}$  和  $y = 0$ , 而其解不唯一。

关于一阶常微分方程的初值问题的解有如下存在唯一性定理。

**定理 2.2.1 (存在唯一性定理)** 对于初值问题式 (2.2.6) 和式 (2.2.7), 如果  $f(x, y)$  在带形区域  $D: \{a \leq x \leq b, -\infty < y < +\infty\}$  中为  $x, y$  的连续函数, 并且函数  $f(x, y)$  对  $y$  满足李普希兹 (Lipschitz) 条件:

$$|f(x, y_1) - f(x, y_2)| \leq L|y_1 - y_2|$$

式中,  $(x, y_1), (x, y_2) \in D$ ,  $L > 0$ , 为 Lipschitz 常数, 则初值问题存在唯一的解。

## 2.3 线性代数的有关概念和定理

### 2.3.1 线性相关和线性无关

**定义 2.3.1 (线性空间)** 给定一个非空集合  $V$ , 和一个数域  $P$ , 对  $V$  中的元素定义两种

代数运算：加法和数量乘法（统称为线性运算），若集合  $V$  对定义的线性运算是封闭的（即  $\forall \alpha, \beta \in V$ ，有  $\alpha + \beta \in V$ ，以及对  $\forall \alpha \in V, k \in P$ ，有  $k\alpha \in V$ ），且线性运算对  $\forall \alpha, \beta, \gamma \in V, k, l \in P$  还具有如下 8 条特性：

加法运算满足：

- (1) 交换律  $\alpha + \beta = \beta + \alpha$
- (2) 结合律  $(\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)$
- (3) 有零元素  $0 \in V$ ，使  $\alpha + 0 = \alpha$
- (4) 对每个元素  $\alpha$  有负元素  $-\alpha$ ，使得  $\alpha + (-\alpha) = 0$

数乘运算满足：

- (5) 有单位元  $1 \in P$ ，使  $1 \cdot \alpha = \alpha$
- (6)  $k(l\alpha) = (kl)\alpha$

对两种运算满足分配律：

- (7)  $k(\alpha + \beta) = k\alpha + k\beta$
- (8)  $(k + l)\alpha = k\alpha + l\alpha$

则称具有线性运算的集合  $V$  为数域  $P$  上的线性空间。

**例 2.3.1** 设  $R$  为实数域，由全体  $n$  维向量组成的集合，在向量的加法和向量与实数的数乘运算下，构成实数域  $R$  上的线性空间（称向量空间），记为  $R^n$ 。由  $m \times n$  矩阵组成的集合，在矩阵的加法运算和矩阵与实数的数乘运算下，构成实数域  $R$  上的一个线性空间（称矩阵空间），记为  $R^{m \times n}$ 。次数小于等于  $n$  的全体多项式，在多项式的加法运算及多项式与实数的乘法运算下，构成数域  $R$  上的一个线性空间（称多项式空间），记为  $H_n$ 。

在实数域  $R$  上定义的线性空间称为实线性空间，在复数域  $C$  上的空间称为复线性空间。

**定义 2.3.2（线性子空间）** 设集合  $V$  是数域  $P$  上的线性空间， $U$  是  $V$  上的一个非空子集，如果  $U$  对线性运算（加法和数乘）是封闭的，即对  $\forall \alpha, \beta \in U, k, l \in P$ ，有  $k\alpha + l\beta \in U$ ，则具有线性运算的集合  $U$  是数域  $P$  上的线性空间，称  $U$  为  $V$  上的一个线性子空间。

例如，当  $m < n$  时，有不高于  $m$  次的多项式全体构成的线性空间  $H_m$  是由不高于  $n$  次的多项式全体构成的线性空间  $H_n$  的子空间。

**定义 2.3.3（线性相关性）** 设集合  $V$  是数域  $P$  上的线性空间， $\alpha_1, \alpha_2, \dots, \alpha_n \in V$ ，如果存在不全为零的数  $k_1, k_2, \dots, k_n \in P$ ，使得

$$k_1\alpha_1 + k_2\alpha_2 + k_3\alpha_3 + \dots + k_n\alpha_n = \sum_{i=1}^n k_i\alpha_i = 0$$

则称  $\alpha_1, \alpha_2, \dots, \alpha_n$  是线性相关的。

否则，对于  $\alpha_1, \alpha_2, \dots, \alpha_n$ ，若满足  $k_1\alpha_1 + k_2\alpha_2 + k_3\alpha_3 + \dots + k_n\alpha_n = 0$ ，可以推出  $k_1 = k_2 = \dots = k_n = 0$ ，则称  $\alpha_1, \alpha_2, \dots, \alpha_n$  是线性无关的。

向量组的线性相关性有如下重要结论：

- (1) 若向量组  $\alpha_1, \alpha_2, \dots, \alpha_n (n \geq 2)$  中有一个向量可以由其他的向量线性表示，那么  $\alpha_1, \alpha_2, \dots, \alpha_n$  必然线性相关。
- (2) 若向量组的部分组线性相关，则整个向量组必然线性相关。
- (3) 线性无关的向量组，它的部分组也线性无关。
- (4) 任意  $n+1$  个  $n$  维向量必然线性相关。

**定义 2.3.4（基、维数与向量的坐标）** 如果向量空间  $V$  中，有  $n$  个线性无关的向量组

$\alpha_1, \alpha_2, \dots, \alpha_n$ , 并且  $V$  中的任意向量  $\alpha$  都可以由向量组  $\alpha_1, \alpha_2, \dots, \alpha_n$  线性表示

$$\alpha = k_1 \alpha_1 + k_2 \alpha_2 + k_3 \alpha_3 + \dots + k_n \alpha_n \quad (2.3.1)$$

则称向量组  $\alpha_1, \alpha_2, \dots, \alpha_n$  为向量空间  $V$  的一个基, 基中所有元素的个数  $n$  称为  $V$  的维数, 并称  $V$  为  $n$  维线性空间, 称式 (2.3.1) 中的  $n$  个数  $k_1, k_2, \dots, k_n$  为向量  $\alpha$  在  $\alpha_1, \alpha_2, \dots, \alpha_n$  下的坐标, 记为  $(k_1, k_2, \dots, k_n)$ , 由  $n$  个线性无关的向量组  $\alpha_1, \alpha_2, \dots, \alpha_n$  生成的线性空间, 用  $V = \text{Span}\{\alpha_1, \alpha_2, \dots, \alpha_n\}$  表示。

**例 2.3.2** 设  $e_i \in R^n$ , 其分量除了第  $i$  个是 1 外, 其余为 0, 即

$$e_1 = (1, 0, 0, \dots, 0)$$

$$e_2 = (0, 1, 0, \dots, 0)$$

...

$$e_n = (0, 0, 0, \dots, 1)$$

则向量组  $e_1, e_2, \dots, e_n$  是线性无关的, 且  $R^n$  中任一向量  $\alpha = (a_1, a_2, \dots, a_n)$  都可以由  $e_1, e_2, \dots, e_n$  线性表示为

$$\alpha = a_1 e_1 + a_2 e_2 + \dots + a_n e_n$$

所以向量组  $e_1, e_2, \dots, e_n$  是  $R^n$  的一个基, 通常称为标准基, 或称  $n$  维单位坐标向量组。

## 2.3.2 方阵及其初等变换

**定义 2.3.5 (矩阵)** 设  $m, n$  为两个自然数,  $P$  是一个数域, 有  $P$  上的  $m \times n$  个数  $a_{ij}$  排成  $m$  行、 $n$  列的阵列

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

称  $A$  为  $P$  上的  $m \times n$  矩阵, 如果数域为实数, 则称为实矩阵, 记为  $A = (a_{ij})_{m \times n}$ , 其中  $a_{ij}$  称为该矩阵的第  $i$  行, 第  $j$  列的元素。当  $m = n$  时, 矩阵  $A = (a_{ij})_{m \times n}$  称为  $n$  阶方阵, 称元素  $a_{11}, a_{22}, \dots, a_{nn}$ , 所在的对角线为  $A$  的主对角线。将矩阵  $A$  的行列互换, 成为  $A$  的转置矩阵, 记为  $A^T$ 。

**定义 2.3.6 (行列式)** 对于  $n$  阶方阵  $A = (a_{ij})_{m \times n}$ , 它的行列式记为  $\det(A)$ , 或

$$\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix}$$

定义

$$\det(A) = \sum_{(j_1, j_2, \dots, j_n)} (-1)^{\tau(j_1, j_2, \dots, j_n)} \quad (2.3.2)$$

式中,  $\Sigma$  是对所有  $n$  级排列求和, 即  $(j_1, j_2, \dots, j_n)$  要取遍所有  $n$  级排列 (共  $n!$  个),  $\tau(j_1, j_2, \dots, j_n)$  为  $(j_1, j_2, \dots, j_n)$  排列逆序数。式 (2.3.2) 的右端称为  $\det(A)$  的展开式。

将  $\det(A)$  中元素  $a_{ij}$  所在的第  $i$  行, 第  $j$  列的元素划去后, 留下来的元素按原来次序所组成的  $n-1$  阶行列式叫做元素  $a_{ij}$  的余子式, 记为  $M_{ij}$ , 而称  $A_{ij} = (-1)^{i+j} M_{ij}$  为元素  $a_{ij}$  的代数余子式。

$n$  阶行列式  $\det(A)$  也等于它的任一行 (列) 各元素与其对应的代数余子式乘积之和, 即对

于任何  $i, j (i, j = 1, 2, 3, \dots, n)$  有

$$\det(\mathbf{A}) = a_{i1}A_{i1} + a_{i2}A_{i2} + \dots + a_{in}A_{in}$$

或

$$\det(\mathbf{A}) = a_{1j}A_{1j} + a_{2j}A_{2j} + \dots + a_{nj}A_{nj}$$

另外, 如果  $\mathbf{A}, \mathbf{B}$  为同阶方阵, 则

$$\det(\mathbf{A} \cdot \mathbf{B}) = \det(\mathbf{A}) \cdot \det(\mathbf{B})$$

**定义 2.3.7 (初等行变换)** 对矩阵实施下列 3 种变换:

- ① 交换第  $i$  行与第  $j$  行的位置 (记为  $r_i \leftrightarrow r_j$ )
- ② 用非零数  $k$  乘第  $i$  行 (记为  $k \times r_i$ )
- ③ 把第  $i$  行的  $k$  倍加到第  $j$  行上去 (记为  $k \times r_i + r_j$ )

分别称为矩阵的第①, ②, ③种初等行变换, 统称为矩阵的初等行变换。

相应地, 可以定义矩阵的初等列变换。矩阵的初等行变换和矩阵的初等列变换, 统称为矩阵的初等变换。

**定义 2.3.8 (初等方阵)** 对单位矩阵  $\mathbf{I}$  (见 2.3.4 节) 实施一次初等行 (列) 变换所得到的矩阵, 称为初等方阵。

因为初等行 (列) 变换只有 3 种, 所以初等方阵也只有 3 种, 它们是:

- ① 交换单位矩阵  $\mathbf{I}$  的第  $i$  行 (列) 与第  $j$  行 (列) 之后的初等方阵, 记为  $p(i, j)$
- ② 用非零数  $k$  乘单位矩阵  $\mathbf{I}$  的第  $i$  行 (列) 后的初等方阵, 记为  $p(i(k))$
- ③ 把单位矩阵  $\mathbf{I}$  的第  $i$  行 (列) 的  $k$  倍加到第  $j$  行 (列) 上之后的初等方阵, 记为  $p(i(k), j)$

容易验证, 3 种初等方阵的行列式都不为 0。

下面的定理说明了初等方阵与初等变换的关系。

**定理 2.3.1** 设  $\mathbf{A}$  是  $n$  阶方阵, 则对  $\mathbf{A}$  实施一次初等行变换, 就相当于在  $\mathbf{A}$  的左边乘上一个相应的  $n$  阶初等方阵, 对  $\mathbf{A}$  实施一次初等列变换, 就相当于在  $\mathbf{A}$  的右边乘上一个相应的  $n$  阶初等方阵。

具体地, 可以用下面的表格来表示:

用矩阵乘法表示初等行变换	用矩阵乘法表示初等列变换
$\mathbf{A} \xrightarrow{r_i \leftrightarrow r_j} p(i, j)\mathbf{A}$	$\mathbf{A} \xrightarrow{c_i \leftrightarrow c_j} \mathbf{A}p(i, j)$
$\mathbf{A} \xrightarrow{k \times r_i} p(i(k))\mathbf{A}$	$\mathbf{A} \xrightarrow{k \times c_i} \mathbf{A}p(i(k))$
$\mathbf{A} \xrightarrow{k \times r_i + r_j} p(i(k), j)\mathbf{A}$	$\mathbf{A} \xrightarrow{k \times c_j + c_i} \mathbf{A}p(i(k), j)$

将方阵经初等行变换时, 方阵的行列式的变化情况可以归纳为:

- (1) 若  $\mathbf{A} \xrightarrow{r_i \leftrightarrow r_j} \mathbf{B}$ , 则  $\det(\mathbf{B}) = -\det(\mathbf{A})$
- (2) 若  $\mathbf{A} \xrightarrow{k \times r_i} \mathbf{B}$ , 则  $\det(\mathbf{B}) = k \det(\mathbf{A})$
- (3) 若  $\mathbf{A} \xrightarrow{k \times r_i + r_j} \mathbf{B}$ , 则  $\det(\mathbf{B}) = \det(\mathbf{A})$

即经第①种初等行变换, 方阵的行列式仅改变符号; 经第②种初等行变换, 行列式变成原行列式的  $k$  倍; 经第③种初等行变换, 行列式不变。对方阵的初等列变换来说, 也有相同的结论。由此可见如果方阵  $\mathbf{A}$  经初等变换后变成了方阵  $\mathbf{B}$ , 则  $\det(\mathbf{A}) \neq 0 \Leftrightarrow \det(\mathbf{B}) \neq 0$ , 即经初等变换后, 方阵的行列式不等于 0 的事实不会改变。

例 2.3.3 证明  $n$  阶范德蒙 (Vandermonde) 行列式 ( $n \geq 2$ )。

$$D_n = \begin{vmatrix} 1 & 1 & \cdots & 1 \\ a_1 & a_2 & \cdots & a_n \\ a_1^2 & a_2^2 & \cdots & a_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ a_1^{n-1} & a_2^{n-1} & \cdots & a_n^{n-1} \end{vmatrix} = \prod_{1 \leq j < i \leq n} (a_i - a_j)$$

式中  $\prod$  为连乘号,  $\prod_{1 \leq j < i \leq n} (a_i - a_j)$  表示所有形如  $(a_i - a_j)$  ( $1 \leq j < i \leq n$ ) 的因子的乘积。

证明 用数学归纳法来证。因为

$$D_2 = \begin{vmatrix} 1 & 1 \\ a_1 & a_2 \end{vmatrix} = a_2 - a_1 = \prod_{1 \leq j < i \leq 2} (a_i - a_j)$$

所以, 当  $n=2$  时结论成立。

假设对于  $D_{n-1}$  成立, 下面证明对于  $D_n$  也成立。

首先, 对  $D_n$  降阶, 具体的做法是: 将  $D_n$  的第 1 列中除  $a_{11}$  元素外的其他元素都化成 0, 然后再把  $D_n$  按第 1 列展开。为此, 先把第  $n-1$  行的  $(-a_1)$  倍加到第  $n$  行上去, 再把第  $n-2$  行的  $(-a_1)$  倍加到第  $n-1$  行上去,  $\cdots$ , 最后把第一行的  $(-a_1)$  加到第 2 行去, 于是得到

$$D_n = \begin{vmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & a_2 - a_1 & a_3 - a_1 & \cdots & a_n - a_1 \\ 0 & a_2(a_2 - a_1) & a_3(a_3 - a_1) & \cdots & a_n(a_n - a_1) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & a_2^{n-2}(a_2 - a_1) & a_3^{n-2}(a_3 - a_1) & \cdots & a_n^{n-2}(a_n - a_1) \end{vmatrix}$$

按第 1 列展开, 然后提出每列的公因子, 得

$$D_n = (a_2 - a_1)(a_3 - a_1) \cdots (a_n - a_1) \begin{vmatrix} 1 & 1 & \cdots & 1 \\ a_2 & a_3 & \cdots & a_n \\ \vdots & \vdots & \vdots & \vdots \\ a_2^{n-2} & a_3^{n-2} & \cdots & a_n^{n-2} \end{vmatrix}$$

上式右端的行列式是一个  $n-1$  阶范德蒙行列式, 由归纳法假设, 它等于  $\prod_{2 \leq j < i \leq n} (a_i - a_j)$ , 于是得

$$D_n = (a_2 - a_1)(a_3 - a_1) \cdots (a_n - a_1) \prod_{2 \leq j < i \leq n} (a_i - a_j)$$

即对  $n$  阶范德蒙行列式也成立, 于是结论对任意阶的范德蒙行列式都成立。

本例说明,  $n$  阶范德蒙行列式不等于零的充要条件是  $a_1, a_2, \cdots, a_n$  互不相等。

### 2.3.3 线性方程组解的存在唯一性

定理 2.3.2 (克莱姆法则) 对于  $n$  个方程,  $n$  个未知变量的线性方程组

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases} \quad (2.3.3)$$

其矩阵形式为  $Ax = b$ , 式中



$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

如果  $\det(A) \neq 0$ ，则方程组 (2.3.3) 有唯一的解

$$x_1 = \frac{\det(A_1)}{\det(A)}, x_2 = \frac{\det(A_2)}{\det(A)}, \dots, x_n = \frac{\det(A_n)}{\det(A)}$$

式中，矩阵  $A_j$  是用  $b$  代换系数矩阵  $A$  的第  $j$  列后所得到的  $n$  阶方阵，即

$$A_j = \begin{pmatrix} a_{11} & \cdots & a_{1(j-1)} & b_1 & a_{1(j+1)} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2(j-1)} & b_2 & a_{2(j+1)} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & \cdots & a_{n(j-1)} & b_n & a_{n(j+1)} & \cdots & a_{nn} \end{pmatrix}$$

如果式 (2.3.3) 中的常向量  $b$  为零向量，则对应的方程组为齐次方程组，否则为非齐次方程组。因为齐次方程组  $Ax=0$  总是有零解，由克莱姆法则可知，对于  $n$  个方程， $n$  个未知量的齐次方程组  $Ax=0$ 。如果  $\det(A) \neq 0$ ，则它只有零解。

**例 2.3.4** 用克莱姆法则求解下列方程组。

$$\begin{cases} 2x_1 + 3x_2 + 5x_3 = 2 \\ x_1 + 2x_2 = 5 \\ 3x_2 + 5x_3 = 4 \end{cases}$$

**解** 由于方程组的系数行列式

$$\det(A) = \begin{vmatrix} 2 & 3 & 5 \\ 1 & 2 & 0 \\ 0 & 3 & 5 \end{vmatrix} = 20 \neq 0$$

因此由克莱姆法则可知，方程组有唯一的解，计算可得

$$\det(A_1) = \begin{vmatrix} 2 & 3 & 5 \\ 5 & 2 & 0 \\ 4 & 3 & 5 \end{vmatrix} = -20$$

$$\det(A_2) = \begin{vmatrix} 2 & 2 & 5 \\ 1 & 5 & 0 \\ 0 & 4 & 5 \end{vmatrix} = 60$$

$$\det(A_3) = \begin{vmatrix} 2 & 3 & 2 \\ 1 & 2 & 5 \\ 0 & 3 & 4 \end{vmatrix} = -20$$

代入式 (2.3.3) 得

$$x_1 = \frac{\det(A_1)}{\det(A)} = \frac{-20}{20} = -1, \quad x_2 = \frac{\det(A_2)}{\det(A)} = \frac{60}{20} = 3, \quad x_3 = \frac{\det(A_3)}{\det(A)} = \frac{-20}{20} = -1$$

从本例可以看出，由于克莱姆法则求解  $n$  阶线性方程组，需要计算  $n+1$  个行列式，计算量非常大，因此在实际计算时，并不实用。不过克莱姆法则有其理论上的价值，例如在不求解的情况下，利用克莱姆法则可分析方程组解的存在唯一性。

### 2.3.4 特殊矩阵

下面介绍一些与数值计算方法有关的特殊方阵及其行列式的计算。

#### (1) 零矩阵

所有元素都是零的  $n \times n$  矩阵, 称为  $n \times n$  零矩阵, 记为  $\mathbf{0}_{n \times n}$ ,  $\mathbf{0}_n$  或  $\mathbf{0}$ , 即

$$\mathbf{0} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}, \quad \det(\mathbf{0}) = 0$$

#### (2) 单位矩阵

主对角线上的元素都是 1, 而其他元素全为零的  $n$  阶方阵称为  $n$  阶单位矩阵, 记为  $\mathbf{E}$  或  $\mathbf{I}$ , 为明确其阶数, 也可以记为  $\mathbf{E}_n$  或  $\mathbf{I}_n$ , 即

$$\mathbf{I} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}, \quad \det(\mathbf{I}) = 1$$

#### (3) 对角矩阵

除对角线以外其他元素全为 0 的  $n$  阶方阵称为对角矩阵  $\mathbf{D}_n$  或  $\mathbf{D}$

$$\mathbf{D} = \begin{pmatrix} d_1 & & & 0 \\ & d_2 & & \\ & & \ddots & \\ 0 & & & d_n \end{pmatrix} \quad \text{或} \quad \mathbf{D} = \text{diag}\{d_1, d_2, \cdots, d_n\}$$

其行列式  $\det(\mathbf{D}) = d_1 d_2 \cdots d_n = \prod_{i=1}^n d_i$ 。

可以证明, 对角矩阵的和、乘积及逆仍为对角矩阵。

#### (4) 上三角矩阵

主对角线下边的元素全为 0, 而主对角线及其上边的元素不全为 0 的  $n$  阶方阵, 称为上三角矩阵, 记为  $\mathbf{U}$  或  $\mathbf{U}_n$ 。

$$\mathbf{U} = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ 0 & & & u_{nn} \end{pmatrix}$$

当  $u_{ii} = 1 (i = 1, 2, \cdots, n)$  时, 称  $\mathbf{U}$  为单位上三角矩阵。当  $u_{ii} = 0 (i = 1, 2, \cdots, n)$  时, 称  $\mathbf{U}$  为严格上三角矩阵。

$$\det(\mathbf{U}) = u_{11} \cdot u_{22} \cdots u_{nn} = \prod_{i=1}^n u_{ii}$$

上三角矩阵的和、乘积及逆仍为上三角矩阵。

#### (5) 下三角矩阵

主对角线上边的元素全为 0, 而主对角线及其下边的元素不全为 0 的  $n$  阶方阵, 称为下三

角矩阵, 记为  $L_n$  或  $L$ 。

$$L = \begin{pmatrix} l_{11} & & & & \\ l_{21} & l_{22} & & & 0 \\ l_{31} & l_{32} & l_{33} & & \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{pmatrix}$$

当  $l_{ii} = 1 (i = 1, 2, \dots, n)$  时, 称  $L$  为单位下三角矩阵。当  $l_{ii} = 0 (i = 1, 2, \dots, n)$  时, 称  $L$  为严格下三角矩阵。

$$\det(L) = l_{11} \cdot l_{22} \cdots l_{nn} = \prod_{i=1}^n l_{ii}$$

下三角矩阵的和、乘积及逆仍为下三角矩阵。

### 2.3.5 方阵的逆及其运算性质

**定义 2.3.9 (逆矩阵)** 设  $A$  为  $n$  阶方阵, 如果存在  $n$  阶方阵  $B$ , 使得  $A \cdot B = B \cdot A = I$ , 则称方阵  $A$  是可逆的, 并称  $B$  为方阵  $A$  的逆矩阵, 记为  $A^{-1}$ , 即  $A^{-1} = B$ 。

设  $A$  为可逆矩阵,  $B$ 、 $C$  都为  $A$  的可逆矩阵, 则

$$AB = BA = I, \quad AC = CA = I$$

于是

$$B = BI = B(AC) = (BA)C = IC = C$$

可见,  $A$  的逆矩阵是唯一的。

**定义 2.3.10 (伴随矩阵)** 设  $A = (a_{ij})_{n \times n}$  为  $n$  阶方阵, 元素  $a_{ij}$  的代数余子式为  $A_{ij} (i, j = 1, 2, \dots, n)$

则称

$$A^* = \begin{pmatrix} A_{11} & A_{21} & \cdots & A_{n1} \\ A_{12} & A_{22} & \cdots & A_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ A_{1n} & A_{2n} & \cdots & A_{nn} \end{pmatrix}$$

为矩阵  $A$  的伴随矩阵。

**定理 2.3.3 (方阵可逆的充要条件)**  $n$  阶方阵  $A$  可逆的充要条件是  $\det(A) \neq 0$ , 且当  $A$  可逆时, 有

$$A^{-1} = \frac{1}{\det(A)} A^*$$

一般将行列式不为 0 的方阵称为非奇异方阵 (否则为奇异方阵)。因此, 由上述定理可知, 方阵  $A$  可逆的充要条件是  $A$  为非奇异矩阵。

因为初等方阵都是可逆的, 且其逆矩阵也是初等方阵:

$$p(i, j)^{-1} = p(i, j)$$

$$p(i(k))^{-1} = p(i(\frac{1}{k}))$$

$$p(i(k), j)^{-1} = p(i(-k), j)$$

用定理 2.3.3 求方阵  $A$  的逆矩阵计算量很大, 一般采用其他方法, 用初等变换法求方阵  $A$  的逆矩阵, 具体的做法是: 在  $n$  阶方阵  $A$  的右边加上一个同阶单位矩阵  $I$ , 得到一个  $n \times 2n$  矩阵  $[A, I]$ , 对它进行一系列初等变换, 直至把  $A$  化成  $I$ , 这时就将  $I$  化成了  $A^{-1}$ , 即

$$[A, I] \xrightarrow{\text{初等行变换}} [I, A^{-1}]$$

例 2.3.5 求矩阵的逆矩阵。

$$A = \begin{pmatrix} 2 & 2 & 3 \\ 1 & -1 & 0 \\ -1 & 2 & 1 \end{pmatrix}$$

$$\begin{aligned} \text{解} \quad [A, I] &= \begin{pmatrix} 2 & 2 & 3 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 1 & 0 \\ -1 & 2 & 1 & 0 & 0 & 1 \end{pmatrix} \xrightarrow{r_1 \leftrightarrow r_2} \begin{pmatrix} 1 & -1 & 0 & 0 & 1 & 0 \\ 2 & 2 & 3 & 1 & 0 & 0 \\ -1 & 2 & 1 & 0 & 0 & 1 \end{pmatrix} \\ &\xrightarrow{\begin{matrix} (-2) \times r_1 + r_2 \\ r_1 + r_3 \end{matrix}} \begin{pmatrix} 1 & -1 & 0 & 0 & 1 & 0 \\ 0 & 4 & 3 & 1 & -2 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \xrightarrow{r_2 \leftrightarrow r_3} \begin{pmatrix} 1 & -1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 4 & 3 & 1 & -2 & 0 \end{pmatrix} \\ &\xrightarrow{(-4) \times r_2 + r_3} \begin{pmatrix} 1 & -1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 & -6 & -4 \end{pmatrix} \xrightarrow{\begin{matrix} r_3 + r_2 \\ (-1) \times r_3 \end{matrix}} \begin{pmatrix} 1 & -1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & -5 & -3 \\ 0 & 0 & 1 & -1 & 6 & 4 \end{pmatrix} \\ &\xrightarrow{r_2 + r_1} \begin{pmatrix} 1 & 0 & 0 & 1 & -4 & -3 \\ 0 & 1 & 0 & 1 & -5 & -3 \\ 0 & 0 & 1 & -1 & 6 & 4 \end{pmatrix} = [A^{-1}] \end{aligned}$$

所以

$$A^{-1} = \begin{pmatrix} 1 & -4 & -3 \\ 1 & -5 & -3 \\ -1 & 6 & 4 \end{pmatrix}$$

逆矩阵具有下列的性质：（设  $A, B$  为同阶可逆矩阵，常数  $k \neq 0$ ）

- ①  $(A^{-1})^{-1} = A$
- ②  $A^T$  可逆，且  $(A^T)^{-1} = (A^{-1})^T$
- ③  $kA$  可逆，且  $(kA)^{-1} = \frac{1}{k}A^{-1}$
- ④  $AB$  可逆，且  $(AB)^{-1} = B^{-1}A^{-1}$
- ⑤  $\det(A^{-1}) = \frac{1}{\det(A)}$

## 2.3.6 矩阵的特征值及其运算性质

定义 2.3.11（矩阵的特征值） 设  $A = (a_{ij})$  是一个  $n$  阶方阵，若存在一个数  $\lambda$  及一个  $n$  维非零列向量

$$X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

使得  $AX = \lambda X$ ，或  $(\lambda I - A)X = 0$ ，则称数  $\lambda$  为方阵  $A$  的一个特征值，称非零列向量  $X$  为方阵  $A$  的对应于特征值  $\lambda$  的特征向量。

由定义可以推出，属于特征值  $\lambda$  的若干特征向量  $X_1, X_2, \dots, X_n$  的任意一个非零的线性组合

$$X = k_1 x_1 + k_2 x_2 + \cdots + k_m x_m \quad (m \neq 0)$$

也属于特征值的  $\lambda$  的特征向量。

$$\begin{aligned} \text{事实上} \quad AX &= A(k_1 x_1 + k_2 x_2 + \cdots + k_m x_m) \\ &= k_1 Ax_1 + k_2 Ax_2 + \cdots + k_m Ax_m \\ &= k_1 \lambda x_1 + k_2 \lambda x_2 + \cdots + k_m \lambda x_m \\ &= \lambda(k_1 x_1 + k_2 x_2 + \cdots + k_m x_m) = \lambda X \end{aligned}$$

所以  $X$  为对应于特征值  $\lambda$  的特征向量。

由  $AX = \lambda X$  可得  $(\lambda I - A)X = 0$ ，又因为齐次方程组有非零解的充要条件是  $\det(\lambda I - A) = 0$ ，所以可得，数  $\lambda$  是矩阵  $A$  的特征值的充要条件是： $\lambda$  是矩阵  $A$  的特征多项式  $f(x) = \det(\lambda I - A)$  的零点。

求  $n$  阶方阵  $A$  的特征值与特征向量的步骤如下。

第一步：计算  $A$  的特征多项式

$$f(\lambda) = \det(\lambda I - A)$$

第二步：求出特征方程  $\det(\lambda I - A) = 0$  的所有根，由于特征方程是一个一元  $n$  次代数方程，若重根则按重数计算，它在实数范围内有  $n$  个根  $\lambda_1, \lambda_2, \cdots, \lambda_n$ ，它们都是  $n$  阶方程  $A$  的特征值。

第三步：对每个特征值  $\lambda_i$ ，求出相应齐次线性方程组  $(\lambda_i I - A)X = 0$  的一个基础解系  $x_1, x_2, \cdots, x_{n-r}$  ( $r$  为其系数矩阵的秩)，并且它是对应于特征值  $\lambda_i$  的线性无关的特征向量，因而它的非零向量线性组合

$$X = k_1 x_1 + k_2 x_2 + \cdots + k_{n-r} x_{n-r} \quad (k_1, k_2, \cdots, k_{n-r} \text{不全为 } 0)$$

即为  $\lambda_i$  的全部特征向量。

**例 2.3.6** 求矩阵  $A = \begin{pmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{pmatrix}$  的特征值与特征向量。

**解** 因为  $\det(\lambda I - A) = (\lambda + 1)^2(\lambda - 5)$ ，所以特征值是  $-1$ （二重）和  $5$ 。

当  $\lambda = -1$  时，由  $(\lambda I - A)X = 0$  得

$$\begin{cases} -2x_1 - 2x_2 - 2x_3 = 0 \\ -2x_1 - 2x_2 - 2x_3 = 0 \\ -2x_1 - 2x_2 - 2x_3 = 0 \end{cases}$$

它的基础解系为

$$x_1 = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}, \quad x_2 = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}$$

因此属于  $-1$  的  $A$  的全部特征值向量为：

$$X = k_1 \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} + k_2 \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix} \quad (\text{其中 } k_1, k_2 \text{ 不全为 } 0)$$

当  $\lambda = 5$  时，由  $(\lambda I - A)X = 0$  得

$$\begin{cases} 4x_1 - 2x_2 - 2x_3 = 0 \\ -2x_1 + 4x_2 - 2x_3 = 0 \\ -2x_1 - 2x_2 + 4x_3 = 0 \end{cases}$$

它的基础解系为

$$\mathbf{x}_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

因此, 属于 5 的  $\mathbf{A}$  的全部特征向量为  $\mathbf{X} = k\mathbf{x}_1 = k \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$

**定义 2.3.12 (谱半径)** 设  $\mathbf{A}$  是  $n$  阶方阵,  $\lambda_i (i=1, 2, \dots, n)$  为  $\mathbf{A}$  的特征值, 则称  $\rho(\mathbf{A}) = \max_{1 \leq i \leq n} |\lambda_i|$  为矩阵  $\mathbf{A}$  的谱半径。

**例 2.3.7** 计算  $\mathbf{A} = \begin{pmatrix} \frac{1}{2} & 0 \\ 16 & \frac{1}{2} \end{pmatrix}$ ,  $\mathbf{B} = \begin{pmatrix} 3 & 6 \\ -2 & 1 \end{pmatrix}$  的谱半径。

**解** 因为  $f(\lambda) = \det(\lambda \mathbf{I} - \mathbf{A}) = \begin{vmatrix} \lambda - \frac{1}{2} & 0 \\ -16 & \lambda - \frac{1}{2} \end{vmatrix} = \left(\lambda - \frac{1}{2}\right)^2 = 0$

所以  $\mathbf{A}$  的特征值  $\lambda_1, \lambda_2 = \frac{1}{2}$ , 所以  $\mathbf{A}$  的谱半径  $\rho(\mathbf{A}) = \frac{1}{2}$ 。

因为  $f(\lambda) = \det(\lambda \mathbf{I} - \mathbf{B}) = \begin{vmatrix} \lambda - 3 & -6 \\ 2 & \lambda - 1 \end{vmatrix} = \lambda^2 - 4\lambda + 15 = 0$

所以  $\mathbf{B}$  的特征值  $\lambda_1 = 2 + \sqrt{11}i, \lambda_2 = 2 - \sqrt{11}i$ , 所以  $\mathbf{B}$  的谱半径  $\rho(\mathbf{B}) = \sqrt{15}$ 。

**定义 2.3.13 (相似矩阵)** 设  $\mathbf{A}, \mathbf{B}$  都是  $n$  阶方阵, 如果存在一个  $n$  阶可逆方阵  $\mathbf{P}$ , 使得

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \mathbf{B}$$

则称方阵  $\mathbf{A}$  与方阵  $\mathbf{B}$  相似, 或  $\mathbf{A}$  相似于  $\mathbf{B}$ , 记为  $\mathbf{A} \sim \mathbf{B}$ 。

相似矩阵有相同的行列式。实际上, 若  $\mathbf{A} \sim \mathbf{B}$ , 则存在可逆矩阵  $\mathbf{P}$ , 使得

$$\mathbf{B} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}$$

所以  $\det(\mathbf{B}) = \det(\mathbf{P}^{-1}\mathbf{A}\mathbf{P}) = \det(\mathbf{P}^{-1}) \cdot \det(\mathbf{A}) \cdot \det(\mathbf{P}) = \det(\mathbf{A})$

$n$  阶方阵  $\mathbf{A}$  的特征值  $\lambda$  和特征向量  $\mathbf{X}$  有下列性质:

- ①  $\mathbf{A}$  与  $\mathbf{A}^T$  具有相同的特征值  $\lambda$ 。
- ② 若  $\mathbf{A}$  可逆, 则  $\mathbf{A}^{-1}$  的特征值为  $\lambda^{-1}$ , 其相应的特征向量也为  $\mathbf{X}$ 。
- ③ 相似矩阵具有相同的特征值, 这是因为设  $\mathbf{A} \sim \mathbf{B}$ , 则

$$\begin{aligned} \det(\lambda \mathbf{I} - \mathbf{B}) &= \det(\lambda \mathbf{I} - \mathbf{P}^{-1}\mathbf{A}\mathbf{P}) = \det(\mathbf{P}^{-1}(\lambda \mathbf{I} - \mathbf{A})\mathbf{P}) \\ &= \det(\mathbf{P}^{-1}) \cdot \det(\lambda \mathbf{I} - \mathbf{A}) \cdot \det(\mathbf{P}) = \det(\lambda \mathbf{I} - \mathbf{A}) \end{aligned}$$

- ④  $\mathbf{AB}$  与  $\mathbf{BA}$  具有相同的特征值, 这是因为

$$\mathbf{A}^{-1}(\mathbf{AB})\mathbf{A} = (\mathbf{A}^{-1}\mathbf{A})(\mathbf{BA}) = \mathbf{BA}$$

即  $\mathbf{AB}$  与  $\mathbf{BA}$  相似。

- ⑤  $\lambda^m$  为矩阵  $\mathbf{A}^m$  的特征值 ( $m$  为自然数), 相应的特征向量也为  $\mathbf{X}$ 。

⑥ 设矩阵  $\mathbf{A}$  的多项式为  $p(\mathbf{A}) = a_0\mathbf{I} + a_1\mathbf{A} + a_2\mathbf{A}^2 + \dots + a_n\mathbf{A}^n$ , 则其特征值为  $p(\lambda) = a_0 + a_1\lambda + a_2\lambda^2 + \dots + a_n\lambda^n$ , 其相应的特征值也为  $\mathbf{X}$ 。

- ⑦ 若  $\mathbf{A}$  的全部特征值为  $\lambda_1, \lambda_2, \dots, \lambda_n$  ( $k$  重特征值算  $k$  个特征值), 则

$$\lambda_1 + \lambda_2 + \cdots + \lambda_n = \sum_{i=1}^n a_{ii}$$

$$\lambda_1 \cdot \lambda_2 \cdots \lambda_n = \det(\mathbf{A})$$

**例 2.3.8** 已知 3 阶矩阵  $\mathbf{A}$  的特征值为 1, 1, 2, 设矩阵  $\mathbf{B} = \mathbf{A}^2 - 3\mathbf{A} + \mathbf{I}$ , 求  $\det(\mathbf{A}), \det(\mathbf{B})$ 。

**解** 设  $\mathbf{A}$  的特征值  $\lambda_1 = 1, \lambda_2 = 1, \lambda_3 = 2$ , 则由特征值的性质得  $\det(\mathbf{A}) = \lambda_1 \cdot \lambda_2 \cdot \lambda_3 = 1 \times 1 \times 2 = 2$ 。

因为  $\mathbf{B} = \mathbf{A}^2 - 3\mathbf{A} + \mathbf{I}$ , 所以  $\mathbf{B}$  的三个特征值分别为

$$\lambda_1^2 - 3\lambda_1 + 1 = 1 - 3 + 1 = -3,$$

$$\lambda_2^2 - 3\lambda_2 + 1 = 1 - 3 + 1 = -3$$

$$\lambda_3^2 - 3\lambda_3 + 1 = 4 - 6 + 1 = -1$$

所以

$$\det(\mathbf{B}) = (-3) \times (-3) \times (-1) = -9$$

## 2.3.7 对称正定矩阵

**定义 2.3.14 (主子阵)** 方阵  $\mathbf{A} = (a_{ij})_{n \times n}$  右上角的各阶方阵

$$\mathbf{A}_1 = (a_{11}), \quad \mathbf{A}_2 = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \quad \mathbf{A}_3 = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}, \quad \cdots, \quad \mathbf{A}_n = \mathbf{A}$$

称为方阵  $\mathbf{A}$  的顺序主子矩阵, 方阵  $\mathbf{A}$  的  $r$  阶主子矩阵的行列式  $\mathbf{D}_r = \det(\mathbf{A}_r)$  ( $r = 1, 2, \cdots, n$ ) 称为  $r$  阶顺序主子式。

**定义 2.3.15 (对称矩阵)** 若方阵  $\mathbf{A} = (a_{ij})_{n \times n}$  满足  $\mathbf{A}^T = \mathbf{A}$  或  $a_{ij} = a_{ji}$  ( $i, j = 1, 2, 3, \cdots$ ), 则称  $\mathbf{A}$  为对称矩阵。

但是对称矩阵的乘积不一定是对称矩阵, 例如

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 8 & 4 \\ 4 & 2 \end{pmatrix}$$

都是对称矩阵, 但  $\mathbf{AB} = \begin{pmatrix} 16 & 8 \\ 32 & 16 \end{pmatrix}$  不是对称矩阵。

**定义 2.3.16 (正交矩阵)** 若方阵  $\mathbf{A} = (a_{ij})_{n \times n}$  满足  $\mathbf{AA}^T = \mathbf{I}$ , 则称  $\mathbf{A}$  为正交矩阵。

例如

$$\begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$$

都为正交矩阵。

正交矩阵具有如下性质:

① 若  $\mathbf{A}$  为正交矩阵, 则  $\mathbf{A}^{-1}, \mathbf{A}^T$  也为正交矩阵。

② 若  $\mathbf{A}$  为正交矩阵, 则  $\det(\mathbf{A}) = \pm 1$ 。

③ 若  $\mathbf{A}, \mathbf{B}$  为同阶正交矩阵, 则  $\mathbf{AB}$  也为正交矩阵。

若  $\mathbf{A}$  为实对称矩阵, 则  $\mathbf{A}$  有如下性质:

①  $\mathbf{A}$  的特征值全为实数。

②  $A$  有  $n$  个线性无关的特征向量。

③ 对应于不同特征值的特征向量必正交。

④ 存在正交矩阵  $P$ , 使  $P^{-1}AP = P^TAP$  为对角矩阵。

**定义 2.3.17 (对称正定矩阵)** 若方阵  $A = (a_{ij})_{n \times n}$  为实对称矩阵, 且对任意的非零列向量  $X$ , 满足

$$X^TAX > 0$$

则称  $A$  为对称正定矩阵, 若  $X^TAX \geq 0$ , 则称  $A$  为半对称正定矩阵。

对称正定矩阵  $A$  具有如下性质:

① 对称正定矩阵  $A$  的对角线元素都是正的, 即  $a_{ii} > 0 (i = 1, 2, \dots, n)$ 。

② 对称矩阵  $A$  正定的充要条件是  $A$  的所有特征值都是正的。

③ 对称矩阵  $A$  正定的充要条件是  $A$  的所有顺序主子式都是正的。

**例 2.3.9** 判断对称矩阵  $A$  是否为正定矩阵。

$$A = \begin{pmatrix} 1 & 2 & -1 \\ 2 & 5 & -1 \\ -1 & -1 & 6 \end{pmatrix}$$

**解** 因为  $A$  的各阶顺序主子式

$$D_1 = 1 > 0, D_2 = \det \begin{pmatrix} 1 & 2 \\ 2 & 5 \end{pmatrix} = 1 > 0, D_3 = \det \begin{pmatrix} 1 & 2 & -1 \\ 2 & 5 & -1 \\ -1 & -1 & 6 \end{pmatrix} = 4 > 0$$

由对称正定矩阵的性质可知,  $A$  为正定矩阵。

## 2.3.8 对角占优矩阵

**定义 2.3.18 (不可约矩阵)** 若  $n$  阶方阵  $A$  存在某些行, 如果将这些行对调, 同时将对应的列对调, 可将  $A$  变为

$$\begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} \quad (2.3.4)$$

式中,  $A_{11}$  为  $r$  阶矩阵,  $A_{22}$  为  $n-r$  阶矩阵, 则称  $A$  为可约矩阵, 否则  $A$  为不可约矩阵。

**例 2.3.10** 判断下列矩阵是否为可约矩阵。

$$A = \begin{pmatrix} 5 & 3 & 1 & 2 \\ 0 & 1 & 0 & 2 \\ 3 & 2 & 1 & 4 \\ 0 & 2 & 0 & 3 \end{pmatrix}, \quad B = \begin{pmatrix} 4 & 2 & & \\ 2 & 4 & 2 & \\ & 2 & 4 & 2 \\ & & 2 & 4 \end{pmatrix}$$

**解** 将矩阵  $A$  的第二行和第三行对调, 同时将第二列和第三列对调, 可将  $A$  变为

$$\begin{pmatrix} 5 & 1 & 3 & 2 \\ 3 & 1 & 3 & 2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 2 & 3 \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}$$

式中,  $A_{11} = \begin{pmatrix} 5 & 1 \\ 3 & 1 \end{pmatrix}, A_{22} = \begin{pmatrix} 1 & 3 \\ 2 & 3 \end{pmatrix}$

所以  $A$  为可约矩阵。对于矩阵  $B$ , 不存在这样的行, 能使其变为式 (2.3.4) 的形式, 所以  $B$  为



不可约矩阵。

定义 2.3.19 (严格对角占优矩阵) 若  $n$  阶方阵  $A = (a_{ij})_{n \times n}$ , 满足

$$|a_{ii}| > \sum_{j=1, j \neq i}^n |a_{ij}| \quad (i=1, 2, \dots, n)$$

则称  $A$  为严格对角占优矩阵。

定义 2.3.20 (不可约对角占优矩阵) 若不可约方阵  $A = (a_{ij})_{n \times n}$ , 满足

$$|a_{ii}| \geq \sum_{j=1, j \neq i}^n |a_{ij}| \quad (i=1, 2, \dots, n)$$

且至少有一个不等式严格成立, 则称  $A$  为不可约对角占优矩阵。

定理 2.3.4 (严格对角占优矩阵的性质) 若  $n$  阶方阵  $A = (a_{ij})_{n \times n}$  是严格对角占优矩阵, 则  $A$  的对角线元素不为 0, 即  $a_{ii} \neq 0 (i=1, 2, \dots, n)$ , 且  $A$  为非奇异矩阵。

定理 2.3.5 (不可约对角占优矩阵的性质) 若  $n$  阶方阵  $A = (a_{ij})_{n \times n}$  为不可约对角占优矩阵, 则  $A$  的对角线元素不为 0, 即  $a_{ii} \neq 0 (i=1, 2, \dots, n)$ , 且  $A$  为非奇异矩阵。

### 2.3.9 向量和连续函数的内积

定义 2.3.21 (向量的内积) 设  $\mathbf{x}, \mathbf{y}$  是  $n$  维实向量

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

则称  $x_1 y_1 + x_2 y_2 + \dots + x_n y_n$  为  $\mathbf{x}$  与  $\mathbf{y}$  的内积, 并记为

$$(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n x_i y_i = \mathbf{x}^T \mathbf{y} = \mathbf{y}^T \mathbf{x}$$

$n$  维向量的内积具有以下基本性质:

- ① 对称性  $(\mathbf{x}, \mathbf{y}) = (\mathbf{y}, \mathbf{x})$
- ② 齐次性  $(k\mathbf{x}, \mathbf{y}) = k(\mathbf{x}, \mathbf{y})$
- ③ 可加性  $(\mathbf{x} + \mathbf{y}, \mathbf{z}) = (\mathbf{x}, \mathbf{z}) + (\mathbf{y}, \mathbf{z})$
- ④ 非负性  $(\mathbf{x}, \mathbf{y}) \geq 0$  当且仅当  $\mathbf{x} = \mathbf{0}$  时,  $(\mathbf{x}, \mathbf{x}) = 0$

定义 2.3.22 (向量正交) 对于两个  $n$  维向量  $\mathbf{x}, \mathbf{y}$ , 若  $(\mathbf{x}, \mathbf{y}) = 0$ , 则称向量  $\mathbf{x}$  与  $\mathbf{y}$  正交或互相垂直。

如果  $n$  维向量  $\mathbf{x}, \mathbf{y}$  为复向量, 则其内积定义为

$$(\mathbf{x} \cdot \mathbf{y}) = x_1 \bar{y}_1 + x_2 \bar{y}_2 + \dots + x_n \bar{y}_n$$

式中,  $\bar{y}_i$  为  $y_i$  的共轭复数。

注意: 复向量没有对称性和齐次性, 但具有可加性和正定性, 即

$$(\mathbf{x}, \mathbf{y}) = (\mathbf{y}, \bar{\mathbf{x}}), \quad (\lambda \mathbf{x}, \mathbf{y}) = \lambda(\mathbf{x}, \mathbf{y}), \quad (\mathbf{x}, \lambda \mathbf{y}) = \bar{\lambda}(\mathbf{x}, \mathbf{y}), \quad \text{式中 } \lambda \text{ 为复数}$$

定义 2.3.23 (连续函数的内积) 设  $f(x), g(x)$  分别为区间  $[a, b]$  内的连续函数, 则定义函数  $f, g$  的内积为

$$(f, g) = \int_a^b f(x)g(x)dx$$

连续函数的内积具有如下性质。

- ① 对称性:  $(f, g) = (g, f)$
- ② 齐次性:  $(\lambda f, g) = \lambda(f, g)$ , 式中  $\lambda$  为实数
- ③ 可加性:  $(f + g, h) = (f, h) + (g, h)$ , 式中  $f, g, h$  为连续函数
- ④ 正定性:  $(f, f) \geq 0$ , 当且仅当  $f(x) = 0$  时,  $(f, f) = 0$

在向量内积和连续函数的内积的基础上可以定义带权内积, 称

$$(x, y) = \sum_i^n \omega_i x_i y_i \quad (\omega_i > 0)$$

为向量  $x, y$  带权  $w_i$  的内积, 称

$$(f, g) = \int_a^b \omega(x) f(x) g(x) dx, \quad \omega(x) \text{ 为非负带权函数}$$

为连续函数  $f, g$  带权  $\omega(x)$  的内积。

**定义 2.3.24 (正交函数序列)** 若函数序列  $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x), \dots$  满足

$$(\varphi_j, \varphi_k) = \int_a^b \omega(x) \varphi_j(x) \varphi_k(x) dx = \begin{cases} 0 & j \neq k \\ A_k \neq 0 & j = k \end{cases}, \text{ 式中 } A_k \text{ 为常数}$$

则称  $\{\varphi_j\}$  为  $[a, b]$  内带权  $\omega(x)$  的正交函数序列。

特别地, 当  $\varphi_j(x)$  为多项式时, 则称  $\{\varphi_j\}$  为  $[a, b]$  内带权  $\omega(x)$  的正交多项式序列。

**定理 2.3.6 (Schawz 不等式)** 若  $x, y$  为  $n$  维向量, 则

$$|(x, y)|^2 \leq (x, x)(y, y)$$

称为 Schawz 不等式。

## 2.3.10 向量、矩阵和连续函数的范数

在讨论函数的误差时, 我们用近似值与精确值之差的大小来度量近似值的精度。类似地, 在讨论线性方程组的近似解向量的误差时, 我们仍用近似解向量与精确解向量的差 (仍是向量) 的“大小”, 即一个实数, 来度量近似值的精度, 为此引入向量与实数之间的特殊函数——向量的范数。

**定义 2.3.25 (向量的范数)** 对于任意的  $n$  维实向量  $x \in R^n$ , 按照一定的规则, 确定一个实值函数  $f(x) = \|x\|$ , 如果  $\|x\|$  满足下面三个性质:

- ① 非负性:  $\|x\| \geq 0$ ,  $\|x\| = 0$ , 当且仅当  $x = 0$ ;
- ② 齐次性: 对任意实数  $k$ , 都有  $\|kx\| = |k| \cdot \|x\|$ ;
- ③ 三角不等式: 对任意的  $x, y \in R^n$ , 都有  $\|x + y\| \leq \|x\| + \|y\|$ 。

则称实值函数  $f(x) = \|x\|$  为向量  $x$  的范数。

**例 2.3.11** 设  $A$  是任一  $n$  阶对称正定矩阵, 证明  $\|x\|_A = (x^T A x)^{\frac{1}{2}}$  是一种向量范数。

**证明** (1) 因为  $A$  对称正定, 故当  $x=0$  时,  $\|x\|_A = 0$ ; 而当  $x \neq 0$  时,  $\|x\|_A = (x^T A x)^{\frac{1}{2}} > 0$

(2) 对任何实数  $c$ , 有

$$\|cx\|_A = \sqrt{(cx)^T A (cx)} = |c| \sqrt{x^T A x} = |c| \cdot \|x\|_A$$

(3) 因为  $A$  正定, 故有分解  $A = LL^T$ , 则

$$\|\mathbf{x}\|_A = (\mathbf{x}^T \mathbf{A} \mathbf{x})^{\frac{1}{2}} = (\mathbf{x}^T \mathbf{L} \mathbf{L}^T \mathbf{x})^{\frac{1}{2}} = ((\mathbf{L}^T \mathbf{x})^T (\mathbf{L}^T \mathbf{x}))^{\frac{1}{2}} = \|\mathbf{L}^T \mathbf{x}\|_2$$

故对任意向量  $\mathbf{x}$  和  $\mathbf{y}$ , 总有

$$\|\mathbf{x} + \mathbf{y}\|_A = \|\mathbf{L}^T (\mathbf{x} + \mathbf{y})\|_2 = \|\mathbf{L}^T \mathbf{x} + \mathbf{L}^T \mathbf{y}\|_2 \leq \|\mathbf{L}^T \mathbf{x}\|_2 + \|\mathbf{L}^T \mathbf{y}\|_2 = \|\mathbf{x}\|_A + \|\mathbf{y}\|_A$$

综上可知,  $\|\mathbf{x}\|_A = (\mathbf{x}^T \mathbf{A} \mathbf{x})^{\frac{1}{2}}$  是一种向量范数。

常用的向量范数有:

$$1\text{-范数} \quad \|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i|$$

$$2\text{-范数} \quad \|\mathbf{x}\|_2 = \left( \sum_{i=1}^n |x_i|^2 \right)^{\frac{1}{2}}$$

$$\infty\text{-范数} \quad \|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|$$

容易验证向量的三个常用范数满足定义 2.3.25 的三个性质。

**例 2.3.12** 计算下列向量的三种常用范数  $\|\mathbf{x}\|_\infty$ 、 $\|\mathbf{x}\|_1$  和  $\|\mathbf{x}\|_2$ 。

$$\mathbf{x} = (1, 0, -1, 2)^T$$

$$\text{解} \quad \|\mathbf{x}\|_1 = |1| + |0| + |-1| + |2| = 4$$

$$\|\mathbf{x}\|_2 = \sqrt{1^2 + 0^2 + (-1)^2 + 2^2} = \sqrt{6}$$

$$\|\mathbf{x}\|_\infty = \max\{1, 0, |-1|, 2\} = 2$$

**定理 2.3.7 (向量范数的等价性)**  $n$  维实向量  $\mathbf{x} \in R^n$  的一切范数都是等价的, 即对  $R^n$  中的任何两种范数  $\|\mathbf{x}\|_\alpha$  与  $\|\mathbf{x}\|_\beta$  存在两个正数  $c_1, c_2 > 0$ , 使得对于任意的向量  $\mathbf{x}$ , 不等式

$$c_1 \|\mathbf{x}\|_\beta \leq \|\mathbf{x}\|_\alpha \leq c_2 \|\mathbf{x}\|_\beta$$

成立。

对于向量的三种常用的范数, 有如下的等价关系:

$$\|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1 \leq \sqrt{n} \|\mathbf{x}\|_2$$

$$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_1 \leq n \|\mathbf{x}\|_\infty$$

$$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \sqrt{n} \|\mathbf{x}\|_\infty$$

向量的等价性说明, 对某个向量  $\mathbf{x}$  来说, 如果它的某种范数小 (或大), 那么它的另两种范数也小 (或大)。

**例 2.3.13** 试证明:

$$(1) \quad \|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_1 \leq n \|\mathbf{x}\|_\infty$$

$$(2) \quad \|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \sqrt{n} \|\mathbf{x}\|_\infty$$

**证明**

$$(1) \quad \|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |x_i| \leq |x_1| + |x_2| + \cdots + |x_n| \leq \|\mathbf{x}\|_1$$

另一方面, 有

$$\|\mathbf{x}\|_1 = |x_1| + |x_2| + \cdots + |x_n| \leq n \max_{1 \leq i \leq n} |x_i| = n \|\mathbf{x}\|_\infty$$

$$(2) \quad \|\mathbf{x}\|_\infty^2 = \max_{1 \leq i \leq n} |x_i|^2 \leq x_1^2 + x_2^2 + \cdots + x_n^2 = \|\mathbf{x}\|_2^2, \text{ 即 } \|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2$$

另一方面, 有

$$\|\mathbf{x}\|_2^2 = x_1^2 + x_2^2 + \cdots + x_n^2 \leq n \max_{1 \leq i \leq n} |x_i|^2 = n \|\mathbf{x}\|_\infty^2$$

故  $\|\mathbf{x}\|_2 \leq \sqrt{n} \|\mathbf{x}\|_\infty$ 。

**定义 2.3.26 (矩阵的范数)** 对于任意的  $n$  阶方阵  $\mathbf{A} \in R^{n \times n}$ , 按照一定的规则, 确定一个实值函数  $f(\mathbf{A}) = \|\mathbf{A}\|$ , 如果  $\|\mathbf{A}\|$  满足如下性质:

- ① 非负性:  $\|\mathbf{A}\| \geq 0$ ,  $\|\mathbf{A}\| = 0$  当且仅当  $\mathbf{A} = \mathbf{0}$ ;
- ② 齐次性: 对于任意的实数  $k$ , 有  $\|k\mathbf{A}\| = |k| \cdot \|\mathbf{A}\|$ ;
- ③ 三角不等式: 对任意的  $\mathbf{A}, \mathbf{B} \in R^{n \times n}$ , 都有  $\|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$ ;
- ④ 矩阵乘法不等式: 对任意的  $\mathbf{A}, \mathbf{B} \in R^{n \times n}$ , 都有  $\|\mathbf{A} \cdot \mathbf{B}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{B}\|$ 。

则称实值函数  $f(\mathbf{A}) = \|\mathbf{A}\|$  为方阵  $\mathbf{A}$  的范数, 记为  $\|\mathbf{A}\|$ 。

**例 2.3.14** 已知  $\mathbf{A} = (a_{ij})_{n \times n}$ , 证明  $\|\mathbf{A}\| = \sum_{i=1}^n \sum_{j=1}^n |a_{ij}|$  是一种矩阵范数。

**证明**

$$(1) \quad \|\mathbf{A}\| = \sum_{i=1}^n \sum_{j=1}^n |a_{ij}| \geq 0 \text{ 且 } \|\mathbf{A}\| = 0 \Leftrightarrow \mathbf{A} = \mathbf{0}$$

(2) 对任意实数  $c$ , 有

$$\|c\mathbf{A}\| = \sum_{i=1}^n \sum_{j=1}^n |ca_{ij}| = |c| \sum_{i=1}^n \sum_{j=1}^n |a_{ij}| = |c| \cdot \|\mathbf{A}\|$$

$$(3) \quad \|\mathbf{A} + \mathbf{B}\| = \sum_{i=1}^n \sum_{j=1}^n |a_{ij} + b_{ij}| \leq \sum_{i=1}^n \sum_{j=1}^n |a_{ij}| + \sum_{i=1}^n \sum_{j=1}^n |b_{ij}| = \|\mathbf{A}\| + \|\mathbf{B}\|$$

$$(4) \quad \begin{aligned} \|\mathbf{AB}\| &= \sum_{i=1}^n \sum_{j=1}^n \left| \sum_{k=1}^n a_{ik} b_{kj} \right| \leq \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n |a_{ik}| |b_{kj}| \\ &\leq \left( \sum_{i=1}^n \sum_{k=1}^n |a_{ik}| \right) \left( \sum_{k=1}^n \sum_{j=1}^n |b_{kj}| \right) = \|\mathbf{A}\| \cdot \|\mathbf{B}\| \end{aligned}$$

故  $\|\mathbf{A}\|$  是一种矩阵范数。

矩阵范数的种类很多, 由于在实际应用中, 矩阵和向量常具有一定的联系。因此要求矩阵范数与向量范数相容, 即满足:

$$\|\mathbf{Ax}\|_p \leq \|\mathbf{A}\|_p \cdot \|\mathbf{x}\|_p \quad \text{或} \quad \frac{\|\mathbf{Ax}\|_p}{\|\mathbf{x}\|_p} \leq \|\mathbf{A}\|_p \quad (p=1, 2, \infty)$$

**定义 2.3.27 (算子范数)** 给定一种向量范数  $\|\mathbf{x}\|_p$  ( $p=1, 2, \infty$ ), 相应地定义了矩阵的一个非负函数  $f(\mathbf{A}) = \|\mathbf{A}\|_p$ , 若有

$$\|\mathbf{A}\|_p = \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{Ax}\|_p}{\|\mathbf{x}\|_p} = \max_{\|\mathbf{x}\|_p=1} \|\mathbf{Ax}\|_p$$

则称  $\|\mathbf{A}\|_p$  为向量范数导出的矩阵范数, 也称  $\|\mathbf{A}\|_p$  为算子范数, 算子范数满足定义 2.3.26 中的 4 个性质。

**定理 2.3.8 (算子范数的性质)** 设  $\mathbf{x}$  为  $n$  维向量,  $\mathbf{A}$  为  $n$  阶方阵, 则算子范数:

①  $\|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|$  称为矩阵  $A$  的行范数。

②  $\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}|$  称为矩阵  $A$  的列范数。

③  $\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)}$  称为矩阵  $A$  的欧几里得范数。

证明 ① 设  $x = (x_1, x_2, \dots, x_n)^T \neq 0, A \neq 0$ , 则

$$\|Ax\|_\infty = \max_{1 \leq i \leq n} \left| \sum_{j=1}^n a_{ij} x_j \right| \leq \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij} x_j| \leq \max_{1 \leq i \leq n} |x_j| \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}| = \|x\|_\infty \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|$$

由算子范数定义知

$$\|A\|_\infty = \max_{x \neq 0} \frac{\|Ax\|_\infty}{\|x\|_\infty} \leq \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|$$

另一方面, 设  $A$  的第  $k$  行元素的绝对值之和达到最大, 即

$$\sum_{j=1}^n |a_{kj}| = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|$$

取一个特定向量  $\eta = (\eta_1, \eta_2, \dots, \eta_n)^T$ , 式中

$$\eta_j = \text{sign}(a_{kj}) = \begin{cases} 1, & \text{当 } a_{kj} \geq 0 \\ -1, & \text{当 } a_{kj} < 0 \end{cases}$$

若按照这种取法, 显然  $\|\eta\|_\infty = 1$ 。从而有

$$\|A\|_\infty = \max_{\|x\|_\infty=1} \|Ax\|_\infty \geq \|A\eta\|_\infty = \max_{1 \leq i \leq n} \left| \sum_{j=1}^n a_{ij} \eta_j \right| = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}| = \sum_{j=1}^n |a_{kj}|$$

因此

$$\max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}| \leq \|A\|_\infty \leq \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|$$

于是

$$\|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|$$

结论②的证明与①类似, 设  $k$  为  $A$  的各列元素绝对值之和达到最大的列, 取向量  $\eta = (\eta_1, \eta_2, \dots, \eta_n)^T$ , 其分量除  $\eta_k = 1$  外, 其余分量均为零, 即可证明结论②。

③ 由于  $A \cdot A^T$  为对称矩阵, 且

$$0 \leq \|Ax\|_2^2 = (Ax)^T Ax = x^T A^T Ax = (A^T Ax \cdot x)$$

因此  $A^T A$  为对称正定矩阵, 其全部特征值  $\lambda_i (i=1, 2, \dots, n)$  均非负。可设  $\lambda_1 \geq \lambda_2 \geq \lambda_3 \cdots \lambda_n \geq 0$ , 由实对称矩阵的性质知, 可设与特征值对应的特征向量  $x_1, x_2, \dots, x_n$  为单位正交向量, 即

$$(x_i, x_j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

所以它们可以作为  $R^n$  的一组基, 对任意的  $x \in R^n$ ,  $x = \sum_{i=1}^n k_i x_i$ 。

如果  $x$  满足  $\|x\|_2 = 1$ , 则有

$$\|x\|_2^2 = (x, x) = \sum_{i=1}^n k_i^2 = 1$$

$$\|A\mathbf{x}\|_2^2 = (A^T A \mathbf{x} \cdot \mathbf{x}) = \sum_{i=1}^n \lambda_i k_i^2 \leq \lambda_1$$

特别地, 取  $\mathbf{x} = \mathbf{x}_1$ , 则有

$$\|A\mathbf{x}_1\|_2^2 = (A^T A \mathbf{x}_1 \cdot \mathbf{x}_1) = \lambda_1$$

所以

$$\|\mathbf{x}\|_2 = \max_{\|\mathbf{x}\|_2=1} \|A\mathbf{x}\|_2 = \sqrt{\lambda_1} = \sqrt{\lambda_{\max}(A^T A)}$$

**例 2.3.15** 设  $A = \begin{pmatrix} 4 & -3 \\ 2 & 1 \end{pmatrix}$ , 求  $\|A\|_\infty$ 、 $\|A\|_1$  及  $\|A\|_2$ 。

**解**  $\|A\|_\infty = \max\{7, 3\} = 7$

$$\|A\|_1 = \max\{6, 4\} = 6$$

因为  $A^T A$  有特征值  $\lambda_1 = 15 + 5\sqrt{5}$ ,  $\lambda_2 = 15 - 5\sqrt{5}$ , 则

$$\|A\|_2 = \sqrt{15 + \sqrt{5}} \approx 5.1167$$

与向量范数相似, 矩阵范数也具有等价性。

**定理 2.3.9 (矩阵范数的等价性)**  $n$  阶方阵  $A$  的一切范数都是等价的, 即对于  $R^{n \times n}$  上的任意两种范数  $\|A\|_\alpha$  与  $\|A\|_\beta$ , 存在两个正数  $c_1, c_2 > 0$ , 使得对于任意的  $n$  阶方阵  $A$ , 不等式

$$c_1 \|A\|_\alpha \leq \|A\|_\beta \leq c_2 \|A\|_\beta$$

成立。

例如:

$$\frac{1}{\sqrt{n}} \|A\|_2 \leq \|A\|_\infty \leq \sqrt{n} \|A\|_2$$

$$\frac{1}{n} \|A\|_\infty \leq \|A\|_1 \leq n \|A\|_\infty$$

**定义 2.3.28 (连续函数的范数)** 对于  $[a, b]$  区间内的任意连续函数  $f$ , 按照一定的规则, 确定某个实值函数  $N(f) = \|f\|$ , 如果  $\|f\|$  满足如下性质:

- ① 非负性:  $\|f\| \geq 0$  且  $\|f\| = 0$ , 当且仅当  $f(x) = 0$
- ② 齐次性: 对任意实数  $k$ , 都有  $\|kf\| = |k| \cdot \|f\|$
- ③ 三角不等式: 对任意的  $f, g \in C[a, b]$ , 都有  $\|f + g\| \leq \|f\| + \|g\|$

则称实值函数  $\|f\|$  为连续函数  $f$  的范数。

常用的连续函数的范数有:

1-范数  $\|f\|_1 = \int_a^b |f(x)| dx$

2-范数  $\|f\|_2 = \sqrt{(f, f)} = \sqrt{\int_a^b \rho(x) f^2(x) dx}$

$\infty$ -范数  $\|f\|_\infty = \max_{x \in [a, b]} |f(x)|$

**例 2.3.16** 设函数  $f(x) = |x-1|$ ,  $f \in C[0, 1]$ ,  $\rho(x) = 1$ , 求  $\|f\|_1$ 、 $\|f\|_2$  和  $\|f\|_\infty$ 。

**解**  $\|f\|_1 = \int_0^1 |x-1| dx = \int_0^1 (1-x) dx = -\int_0^1 (x-1) dx = -\frac{1}{2} (x-1)^2 \Big|_0^1 = \frac{1}{2}$

$$\|f\|_2 = \sqrt{\int_0^1 |(x-1)|^2 dx} = \sqrt{\frac{1}{3}} = \frac{\sqrt{3}}{3}$$

$$\|f\|_\infty = 1$$

### 2.3.11 向量序列与矩阵序列的极限

**定义 2.3.29** 设有  $n$  维向量序列  $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(k)}, \dots$ , 其中  $\mathbf{x}^{(k)} = (x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)})^T$ , 如果存在  $n$  维向量  $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$ , 使得  $\lim_{k \rightarrow \infty} x_i^{(k)} = x_i$  ( $i=1, 2, \dots, n$ ), 则称向量序列  $\{\mathbf{x}^{(k)}\}_0^\infty$  收敛于向量  $\mathbf{x}$ , 记为  $\lim_{k \rightarrow \infty} \mathbf{x}^{(k)} = \mathbf{x}$ 。

**定理 2.3.10**  $n$  维向量序列  $\{\mathbf{x}^{(k)}\}_0^\infty$  收敛于  $n$  维向量  $\mathbf{x}$ , 当且仅当  $\lim_{k \rightarrow \infty} \|\mathbf{x}^{(k)} - \mathbf{x}\| = 0$ 。

**证明** 由定义 2.3.29 知,  $\{\mathbf{x}^{(k)}\}_0^\infty$  收敛于  $\mathbf{x}$ , 即

$$\lim_{k \rightarrow \infty} x_i^{(k)} = x_i \quad (i=1, 2, \dots, n)$$

对于任意的  $i(1 \leq i \leq n)$  有

$$0 \leq |x_i^{(k)} - x_i| \leq \max_{1 \leq j \leq n} |x_j^{(k)} - x_j| = \|\mathbf{x}^{(k)} - \mathbf{x}\|_\infty$$

因此

$$\lim_{k \rightarrow \infty} \|\mathbf{x}^{(k)} - \mathbf{x}\|_\infty = 0$$

由向量范数的等价性, 对于任意定义的向量范数, 都有

$$\lim_{k \rightarrow \infty} \|\mathbf{x}^{(k)} - \mathbf{x}\| = 0$$

由向量范数的等价性, 若向量序列在某种范数下收敛, 则它在任何范数下都是收敛的, 同理可定义矩阵序列的极限。

**定义 2.3.30** 设有  $n$  阶方阵序列  $\mathbf{A}^{(0)}, \mathbf{A}^{(1)}, \dots, \mathbf{A}^{(k)}, \dots$ , 其中  $\mathbf{A}^{(k)} = (a_{ij}^{(k)})_{n \times n}$ , 若有矩阵  $\mathbf{A} = (a_{ij})_{n \times n}$ , 使得  $\lim_{k \rightarrow \infty} a_{ij}^{(k)} = a_{ij}$  ( $i, j=1, 2, \dots, n$ ), 则称矩阵序列  $\{\mathbf{A}^{(k)}\}_0^\infty$  收敛于  $\mathbf{A}$ , 记为  $\lim_{k \rightarrow \infty} \mathbf{A}^{(k)} = \mathbf{A}$ 。

可以证明  $\lim_{k \rightarrow \infty} \mathbf{A}^{(k)} = \mathbf{A}$ , 当且仅当  $\lim_{k \rightarrow \infty} \|\mathbf{A}^{(k)} - \mathbf{A}\| = 0$ 。

**定理 2.3.11** 设  $\mathbf{A}$  为  $n$  阶方阵, 由  $\mathbf{A}$  的各次幂所组成的矩阵序列  $\mathbf{I}, \mathbf{A}, \mathbf{A}^2, \dots, \mathbf{A}^k, \dots$  收敛于零矩阵, 即  $\lim_{k \rightarrow \infty} \mathbf{A}^k = \mathbf{0}$  的充分必要条件是  $\rho(\mathbf{A}) < 1$ , 其中  $\rho(\mathbf{A}) = \max_k |\lambda_k|$  为  $\mathbf{A}$  的谱半径 ( $\lambda_k$  为  $\mathbf{A}$  的特征值)。

**定理 2.3.12** 若  $\mathbf{A}$  为  $n$  阶方阵, 则

$$\textcircled{1} \quad \rho(\mathbf{A}) \leq \|\mathbf{A}\|$$

$$\textcircled{2} \quad \text{若 } \mathbf{A}^T = \mathbf{A}, \text{ 则 } \|\mathbf{A}\|_2 = \rho(\mathbf{A})$$

**证明** 因为  $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$

所以对于任意的非零向量  $\mathbf{x}$ , 有

$$|\lambda| \cdot \|\mathbf{x}\| = \|\lambda\mathbf{x}\| = \|\mathbf{A}\mathbf{x}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{x}\|$$

从而得  $|\lambda| \leq \|\mathbf{A}\|$ , 即  $\rho(\mathbf{A}) \leq \|\mathbf{A}\|$ 。

$$\text{当 } \mathbf{A}^T = \mathbf{A} \text{ 时, } \|\mathbf{A}\|_2 = \sqrt{\lambda_{\max}(\mathbf{A}^T \mathbf{A})} = \sqrt{\lambda_{\max}^2(\mathbf{A})} = \rho(\mathbf{A})$$

## 本章小结

本章介绍了计算方法需要的数学基本知识,并给出少量例题,其目的是方便学生复习和查阅,以便更好地学习和掌握后续章节的知识。

极限是各类迭代法的基础;夹逼定理用于范数的等价性;单调有界定理用于牛顿迭代法的全局收敛性;一致连续性定理用于分段低次插值和高斯积分的稳定性;零点定理用于方程求根的存在性;罗尔定理用于证明插值余项定理;积分加权平均值定理用于求辛普生积分公式的余项;线性代数的知识,如特征值、谱半径和范数的概念,用于解线性方程组迭代法的收敛性判断以及矩阵特征值的求解;矩阵的初等变换用于解线性方程组的消去法;范德蒙行列式用于证明插值多项式的存在唯一性和插值型求积公式的代数精度。

## 习题 2

2.1 试证  $f(x) = x^3 - x - 1$  在  $[1, 2]$  内有唯一的解。

2.2 设  $f(x) = e^{x^2}$ , 试求  $f(x)$  在  $x_0 = 1$  的泰勒展开的 3 次多项式。

2.3 求向量  $\mathbf{x} = (1, -2, 3, 4, 8, -1)^T$  的 1-范数、2-范数和  $\infty$ -范数。

2.4 求矩阵  $A = \begin{pmatrix} 1 & 2 \\ -3 & 4 \end{pmatrix}$  的各种范数。

2.5 (1) 设  $\mathbf{x} = [3 \ -1 \ 5 \ 8]^T$ , 求  $\|\mathbf{x}\|_1$ 、 $\|\mathbf{x}\|_\infty$  和  $\|\mathbf{x}\|_2$ ;

(2) 已知  $A = \begin{pmatrix} 4 & -3 \\ -1 & 6 \end{pmatrix}$ , 求  $\|A\|_1$ 、 $\|A\|_\infty$  和  $\|A\|_2$ 。

2.6 记  $\|\mathbf{x}\|_p = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{\frac{1}{p}}$ , 式中  $\mathbf{x} = (x_1, x_2, \cdots, x_n)^T$ , 证明  $\lim_{p \rightarrow \infty} \|\mathbf{x}\|_p = \|\mathbf{x}\|_\infty$ 。

2.7 设  $A = \begin{pmatrix} 2 & 1 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 2 \end{pmatrix}$ , 求  $A$  的特征值和谱半径。

2.8 设  $A = \begin{pmatrix} 3 & -a & -b \\ -a & 3 & -a \\ -b & -a & 3 \end{pmatrix}$ , 试求使  $A$  为严格对角占优矩阵的  $a$  和  $b$  的取值范围。

2.9 设  $A = \begin{pmatrix} 1 & 0 & 0 \\ 2 & -1 & 0 \\ 0 & -2 & \sqrt{5} \end{pmatrix}$ , 求  $\|A\|_1$ 、 $\|A\|_2$  和  $\|A\|_\infty$ 。

2.10 设  $f(x) = (x-1)^3$ , 求  $\|f\|_1$ 、 $\|f\|_2$  和  $\|f\|_\infty$ 。

2.11 求  $A = \begin{pmatrix} 3 & -2 & 0 & -1 \\ 0 & 2 & 2 & 1 \\ 1 & -2 & 3 & -2 \\ 0 & 1 & 2 & 1 \end{pmatrix}$  的逆矩阵。



2.12 判定函数  $\varphi_0(x)=1, \varphi_1(x)=x, \varphi_2(x)=x^2-\frac{1}{3}$  在  $[-1,1]$  内两两正交, 并求  $\varphi_3 \in H_3$  使其在  $[-1,1]$  内与  $\varphi_0, \varphi_1, \varphi_2$  正交。

2.13 构造  $[0,1]$  内带权  $\rho(x)=\ln \frac{1}{x}$  的前 3 个正交多项式  $\varphi_0(x)$ 、 $\varphi_1(x)$  和  $\varphi_2(x)$ 。

2.14 (1)  $\mathbf{x}=(3,0,-4,12)^T$ , 求  $\|\mathbf{x}\|_1$ 、 $\|\mathbf{x}\|_2$  和  $\|\mathbf{x}\|_\infty$ 。

(2)  $A=\begin{pmatrix} 1 & -1 \\ 2 & 3 \end{pmatrix}$ , 求  $\|A\|_1$ 、 $\|A\|_2$ 、 $\|A\|_\infty$  和  $\|A\|_F$ 。

(3)  $A=\begin{pmatrix} 1 & 1 \\ -5 & 1 \end{pmatrix}$ , 求  $A$  的谱半径  $\rho(A)$ 。

2.15 设  $A=\begin{pmatrix} 1 & 1 \\ 2 & 5 \end{pmatrix}, \mathbf{x}=\begin{pmatrix} 2 \\ 3 \end{pmatrix}$ , 求  $\|A\|_1$ 、 $\|A\|_2$ 、 $\|A\|_\infty$ 、 $\|A\mathbf{x}\|_1$ 、 $\|A\mathbf{x}\|_2$  和  $\|A\mathbf{x}\|_\infty$ 。

## 第3章 方程求根



### 学习要点

- (1) 方程求根的三个基本问题：根的存在性、根的分布、根的精确化。
- (2) 二分法：将隔离区间二分，根据函数的符号变化逐步缩短隔离区间。
- (3) 迭代法：将方程  $f(x)=0$  等价转换为  $x=\varphi(x)$ ，并构造迭代公式  $x_{k+1}=\varphi(x_k)$ 。
- (4) 牛顿迭代法：
$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)}$$
- (5) 迭代法的收敛性、收敛速度和迭代法的改善。



### 教学建议

在本章介绍的方程求根的各种方法中，迭代法和牛顿法是重点，要求学生掌握用迭代法求方程的基本思想、几何意义并理解收敛性定理的前提和结论。会构造方程求根的迭代公式并能进行收敛性判断。牛顿法是特殊的迭代法，具有收敛速度快和应用广泛的特点，要求学生掌握牛顿法及其收敛性。Steffensen 迭代法和牛顿法的双重根情形可作为选学内容，根据学生的知识水平灵活安排。建议学时为 4~8 学时。

### 3.1 引言

在科学计算中，常常需要求高次代数方程或超越方程  $f(x)=0$  的解。方程  $f(x)=0$  的解通常称为方程的根，或称为函数  $f(x)$  的零点。

如果  $f(x)$  为  $n$  次多项式

$$f(x)=a_n x^n+a_{n-1} x^{n-1}+\cdots+a_1 x+a_0 \quad (a_n \neq 0)$$

则称相应的方程  $f(x)=0$  为  $n$  次代数方程。

如果  $f(x)$  中含有三角函数、对数函数等其他超越函数，如：

$$f(x)=e^{-x}+\ln x-\sin \frac{\pi x}{2}$$

则称相应的方程  $f(x)=0$  为超越方程。

方程的根可能是实数，也可能是复数，分别称为方程的实根和复根。本章主要介绍方程实根的求法。

如果对于  $x^*$  有  $f(x^*)=0$ ，但  $f'(x^*) \neq 0$ ，则称  $x^*$  为方程  $f(x)=0$  的单根。如果有

$$f(x^*)=f'(x^*)=\cdots=f^{(k-1)}(x^*)=0, \text{ 但 } f^{(k)}(x^*) \neq 0$$

则称  $x^*$  为  $f(x)=0$  的  $k$  重根。此时函数  $f(x)$  可以表示为

$$f(x)=(x-x^*)^k g(x), \quad g(x^*) \neq 0$$

当  $k$  为奇数时,  $f(x)$  在点  $x^*$  处变号; 当  $k$  为偶数时,  $f(x)$  在点  $x^*$  处不变号。

在大多数情况下, 对于高于 4 次的代数方程及超越方程没有精确的求根公式。一般而言, 必须用数值计算方法求它的近似解。事实上, 实际应用中也不一定需要得到根的精确表达式, 只要得到满足一定精度要求的近似解就可以了。

求解方程  $f(x)=0$  的根一般包括下面三个基本问题。

### (1) 根的存在性

根的存在性要回答方程有没有根? 如果有根, 有几个根?

对于代数方程, 由代数学基本定理可知, 其根 (实根和复根) 的个数与其次数相同。对于超越方程, 情况比较复杂, 方程可能有解, 也可能无解。如果有解, 其解可能是一个或几个, 也可能是无穷多个。这个问题已超过了本书的讨论范围。

### (2) 根的分布

根的分布是指先求出方程的有根区间, 然后把有根区间分成若干子区间, 每个子区间或者没有根, 或者只有一个根。只有一个根的子区间称为根的隔离区间, 在隔离区间内的任一点都可看成该根的一个近似值。

求出方程的有根区间, 然后就可以用图解法和试验法, 在有根区间内进行根的隔离。

图解法的思想是, 画出  $y=f(x)$  的粗略图形, 以便确定曲线  $y=f(x)$  与  $x$  轴交点的粗略位置, 从而确定根的隔离区间。

试验法的思想是, 求出  $f(x)$  在若干点上的函数值, 观察函数值符号的变化情况, 从而确定根的隔离区间。具体做法是: 从有根区间  $[a, b]$  的左端点  $a_0 = a$  开始, 按一定的步长  $h$  (如  $h = \frac{b-a}{n}$ ), 逐步向右“搜索”, 即检查节点  $x_k = a + kh$  ( $k = 0, 1, 2, \dots, n$ ) 上函数值  $f(x_k)$  符号, 若  $f(x_{k-1}) \cdot f(x_k) \leq 0$ , 则所求根  $x^*$  必在  $x_{k-1}$  与  $x_k$  之间, 从而确定一个根的隔离区间  $[x_{k-1}, x_k]$ 。

### (3) 根的精确化

当求出方程  $f(x)=0$  的一个根的隔离区间后, 可以取根的隔离区间内的任一值作为方程的近似值, 然后设法将根的近似值进一步精确化, 直到满足精度要求为止。本章后续几节将要介绍几种根的精确化方法, 如二分法、迭代法、牛顿法等, 这些方法对代数方程和超越方程都是适用的。

## 3.2 二分法

设函数  $f(x)$  在区间  $[a, b]$  内单调连续, 且  $f(a) \cdot f(b) < 0$ , 根据连续函数的性质可知方程  $f(x)=0$  在  $[a, b]$  内一定有唯一的实根。为确定起见, 不妨设  $[a, b]$  为隔离区间, 即方程  $f(x)=0$  在  $[a, b]$  内有唯一的实根  $x^*$ 。

用二分法求方程  $f(x)=0$  的实根  $x^*$  的近似值, 其主要思想是: 将含有根  $x^*$  的隔离区间二分, 通过判断二分点与边界点函数值的符号, 逐步对半缩小隔离区间, 直到缩小到满足精度要求为止, 然后取最后二分区间的中点为根  $x^*$  的近似值。其具体步骤如下:

首先把区间  $[a, b]$  二等分, 取区间  $[a, b]$  的中点  $x_0 = \frac{1}{2}(a+b)$ , 计算函数值  $f(x_0)$ 。如果

$f(x_0)=0$ ，则求得实根  $x^*=\frac{1}{2}(a+b)$ ；否则， $f(x_0)$  或者与  $f(a)$  异号，或者与  $f(b)$  异号。

若  $f(a) \cdot f(x_0) < 0$ ，则说明根在区间  $[a, x_0]$  内，这时取  $a_1 = a, b_1 = x_0 = \frac{a+b}{2}$ 。

若  $f(x_0) \cdot f(b) < 0$ ，则说明根在区间  $[x_0, b]$  内，这时取  $a_1 = x_0 = \frac{a+b}{2}, b_1 = b$ 。

不论是哪种情况，新的隔离区间  $[a_1, b_1]$  的长度仅为原来隔离区间  $[a, b]$  的一半，如图 3.2.1 所示。在新的隔离区间  $[a_1, b_1]$  上重复上述二分过程，则会得到下面的隔离区间序列

$$[a, b] \supset [a_1, b_1] \supset [a_2, b_2] \supset \cdots \supset [a_k, b_k] \supset \cdots$$

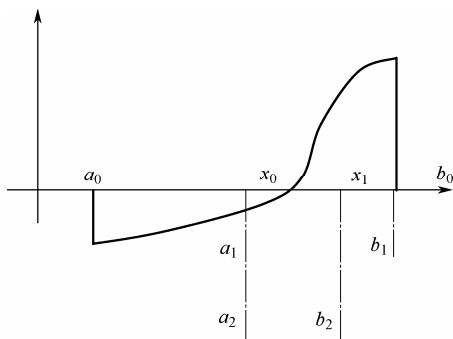


图 3.2.1 二分法示意图

其中，每个区间均为前一个区间的一半，经过  $k$  次二分后，可得新的隔离区间  $[a_k, b_k]$ ，其长度为

$$b_k - a_k = \frac{1}{2^k}(b - a) \quad (3.2.1)$$

如果用上述步骤无限次地二分区间  $[a, b]$  ( $k \rightarrow \infty$ )，则隔离区间必定收缩为一点  $x^*$ ，显然，该点就是方程的根。

在实际计算中，没有必要也不可能进行无限次的二分区间  $[a, b]$ ，只要求得满足预定精度的近似值即可。如果取隔离区间  $[a_k, b_k]$  的中点  $x_k = \frac{1}{2}(a_k + b_k)$  为  $x^*$  的近似值，则在上述二分过程中，可得到以  $x^*$  为极限的根的近似值序列

$$x_0, x_1, x_2, \cdots, x_k, \cdots$$

由于

$$|x^* - x_k| \leq \frac{1}{2}(b_k - a_k) = \frac{1}{2^{k+1}}(b - a) \quad (3.2.2)$$

对于预先给定的精度  $\varepsilon > 0$ ，只要

$$\begin{aligned} \left| \frac{1}{2^{k+1}}(b - a) \right| &< \varepsilon \\ \left| \frac{1}{2^k} \right| &< \frac{2\varepsilon}{b - a}, 2^k > \frac{b - a}{2\varepsilon} \end{aligned}$$

即当  $k > \frac{\ln(b - a) - \ln 2\varepsilon}{\ln 2}$  时，则有

$$|x^* - x_k| < \varepsilon$$

此时,  $x_k$  就是满足精度要求的近似值。

**例 3.2.1** 用二分法求方程  $f(x) = \sin x - \frac{x^2}{4} = 0$  的非零实根的近似值, 使误差不超过  $10^{-2}$ 。

**解** 由于曲线  $y = \sin x$  与  $y = \frac{x^2}{4}$  除原点外有且只有一个交点, 其横坐标介于  $1.5 \sim 2$  之间, 因此, 方程  $f(x) = 0$  在  $[1.5, 2]$  内只有唯一的非零实根  $x^*$ 。

由于  $\frac{2-1.5}{2^{k+1}} \leq 10^{-2}$ , 即  $2^{k+2} \geq 10^2$ , 因此可以确定所需的二分次数  $k=5$ 。

计算结果见表 3.2.1。

表 3.2.1 例 3.2.1 的计算结果

$k$	$a_k$	$b_k$	$x_k$	$f(x_k)$
0	1.5	2	1.75	0.218 361
1	1.75	2	1.875	0.075 179 5
2	1.875	2	1.9375	-0.004 962 28
3	1.875	1.9375	1.906 25	0.035 813 793
4	1.906 25	1.9375	1.921 875	0.015 601 413
5	1.921 875	1.9375	1.929 687 5	0.005 363 40

因此,  $x^* \approx 1.93$ 。

**例 3.2.2** 用二分法求方程  $x^3 - x - 1 = 0$  在  $[1, 2]$  内的近似值, 精度为  $10^{-3}$ , 试问: 要达到此精度至少二分多少次?

**解**  $f(x) = x^3 - x - 1, f(1) = -1, f(2) = 5$ , 且  $f'(x) = 3x^2 - 1 > 0$  单调, 故方程在  $[1, 2]$  内有唯一的实根, 由二分法误差估计知  $|x^* - x_n| \leq \frac{b-a}{2^{n+1}} < 10^{-3}$ , 这里  $a=1, b=2$ , 由此可知, 要使  $\frac{1}{2^{n+1}} < 10^{-3}$ , 则  $(n+1)\lg 2 \geq 3, n \geq \frac{3}{\lg 2} - 1$ , 因此, 当  $n=9$  时达到所要求精度。各次计算结果见表 3.2.2。

表 3.2.2 例 3.2.2 的计算结果

$n$	$a_n$	$b_n$	$x_n$	$f(x_n)$ 的符号
0	1	2	1.5	+
1	1	1.5	1.25	-
2	1.25	1.5	1.375	+
3	1.25	1.375	1.3125	-
4	1.3125	1.375	1.3438	+
5	1.3125	1.3438	1.3281	+
6	1.3125	1.3281	1.3203	-
7	1.3203	1.3281	1.3242	-
8	1.3242	1.3281	1.3262	+
9	1.3242	1.3262	1.3252	+

可见,  $x_9 = 1.3252$  为方程的近似解。

由例 1.1.1 知, 方程  $x^3 - x - 1 = 0$  在  $[1, 2]$  的精确值为

$$x = \sqrt[3]{\frac{1}{2} + \sqrt{\frac{23}{108}}} + \sqrt[3]{\frac{1}{2} - \sqrt{\frac{23}{108}}} = 1.324\ 717\ 958 \dots$$

**例 3.2.3** 证明  $1 - x - \sin x = 0$  在  $[0, 1]$  内仅有一个根, 使用二分法求误差不大于  $\frac{1}{2} \times 10^{-4}$  的根

要二分多少次?

**解**  $f(x) = 1 - x - \sin x$ , 则  $f(0) = 1 > 0$ ,  $f(1) = -\sin 1 < 0$ 。又因为,  $f'(x) = -1 - \cos x < 0$ ,  $x \in [0, 1]$ , 故  $f(x)$  在  $[0, 1]$  内单调递减。因此  $f(x)$  在  $[0, 1]$  内有且仅有一个根。使用二分法时, 误差限为

$$|x_k - x^*| \leq \frac{1}{2^{k+1}}(b-a) = \frac{1}{2^{k+1}} < \frac{1}{2} \times 10^{-4}$$

解得

$$\begin{aligned} 2^k &\geq 10^4 \\ k &\geq \frac{4 \lg 10}{\lg 2} = 13.2877 \end{aligned}$$

所以需二分 14 次。

二分法的优点是: 方法简单, 编程容易, 对函数  $f(x)$  的性质要求不高, 只要求  $f(x)$  在根的隔离区间内连续, 并且在两端点处的函数值异号。其收缩速度与公比为  $\frac{1}{2}$  的等比级数的收缩速度相同。

二分法的缺点是: 只能求实函数的实根, 不能求方程的复根及偶数重根。

如图 3.2.2 所示为二分法的算法框图: 算法中  $a, b$  分别表示各有根区间的左、右端点,  $k$  用于记录二分次数,  $N$  为最大二分次数,  $\varepsilon_1, \varepsilon_2$  为允许误差, 当  $|f(x)| < \varepsilon_1$  或  $b - a < \varepsilon_2$  时计算终止。

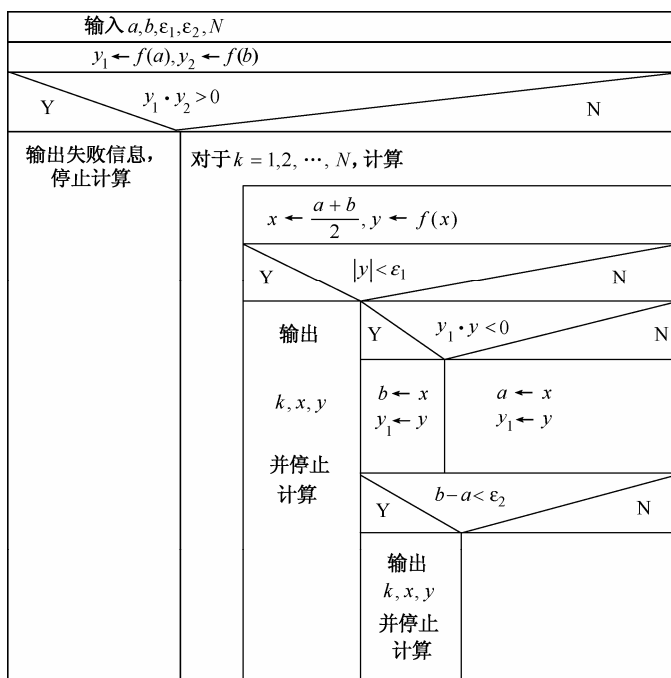


图 3.2.2 二分法的算法框图

## 3.3 迭代法

### 3.3.1 不动点迭代

**例 3.3.1** 求方程  $f(x) = x^3 - x - 1 = 0$  的实根。

**解** 由于  $f(x)$  是在  $[1, 2]$  内的单调函数，且  $f(1) = -1 < 0, f(2) = 5 > 0$ ，因此方程  $f(x) = 0$  在  $[1, 2]$  内有唯一的实根。

将方程  $f(x) = 0$  转换成两种等价的形式

$$x = \varphi_1(x) = \sqrt[3]{x+1}, \quad x = \varphi_2(x) = x^3 - 1$$

取初值  $x_0 = 1.5$ ，代入  $\varphi_1(x)$  中得新值  $x_1 = 1.35721$ ，继续代入上式得到一系列的近似值，见表 3.3.1。

表 3.3.1 例 3.3.1 的计算结果

$k$	$x_k$	$k$	$x_k$	$k$	$x_k$
0	1.5	3	1.325 88	6	1.324 73
1	1.357 21	4	1.324 92	7	1.324 72
2	1.330 86	5	1.324 76	8	1.324 72

由表 3.3.1 可以看出  $x_7 = x_8$ ，这时可以认为  $x_8$  为方程的一个近似根。这种求方程的方法称为迭代法。

但是，假若在例 3.3.1 中，将  $f(x) = 0$  转换为  $x = \varphi_2(x) = x^3 - 1$ ，并建立迭代公式  $x_{k+1} = x_k^3 - 1$ ，仍取  $x_0 = 1.5$ ，则  $x_1 = 2.375, x_2 = 12.3965, x_3 = 1904.01 \dots$  显然这一迭代过程是发散的。

一般地，为了求一元非线性方程

$$f(x) = 0 \quad (3.3.1)$$

的根，可以先将其转换为如下的等价形式

$$x = \varphi(x) \quad (3.3.2)$$

式中连续函数  $\varphi(x)$  称为迭代函数，并使两个方程具有相同的解，然后构造迭代公式

$$x_{k+1} = \varphi(x_k), \quad k = 0, 1, 2, \dots \quad (3.3.3)$$

对于给定的初值  $x_0$ ，由式 (3.3.3) 可产生一个迭代序列  $\{x_k\}_{k=0}^{\infty}$ 。

如果有  $\lim_{k \rightarrow \infty} x_k = x^*$ ，由于  $\varphi(x)$  连续，则

$$x^* = \lim_{k \rightarrow \infty} x_{k+1} = \lim_{k \rightarrow \infty} \varphi(x_k) = \varphi(\lim_{k \rightarrow \infty} x_k) = \varphi(x^*)$$

因此， $x^*$  是式 (3.3.2) 的解，由等价性可知， $x^*$  也是式 (3.3.1) 的解。

此时，称  $x^*$  为  $\varphi(x)$  的不动点，迭代公式 (3.3.3) 称为收敛的，即不动点迭代法。由于在式 (3.3.3) 中， $x_{k+1}$  仅由  $x_k$  决定，因此式 (3.3.3) 又称为单步迭代法， $x_k$  称为方程根的第  $k$  次近似值。如果迭代序列  $\{x_k\}$  的极限不存在，则称迭代公式 (3.3.3) 是发散的。

因此，在使用迭代法求方程根的近似值时，首先要考虑的问题是：如何选取迭代函数  $\varphi(x)$ ，使迭代公式  $x_{k+1} = \varphi(x_k)$  收敛。

### 3.3.2 迭代法的收敛性

为了研究迭代法的收敛性, 首先介绍迭代法的几何意义. 从几何上讲, 求方程  $x = \varphi(x)$  的根, 即求直线  $y = x$  与曲线  $y = \varphi(x)$  的交点  $P$  的横坐标  $x^*$ , 如图 3.3.1 所示.

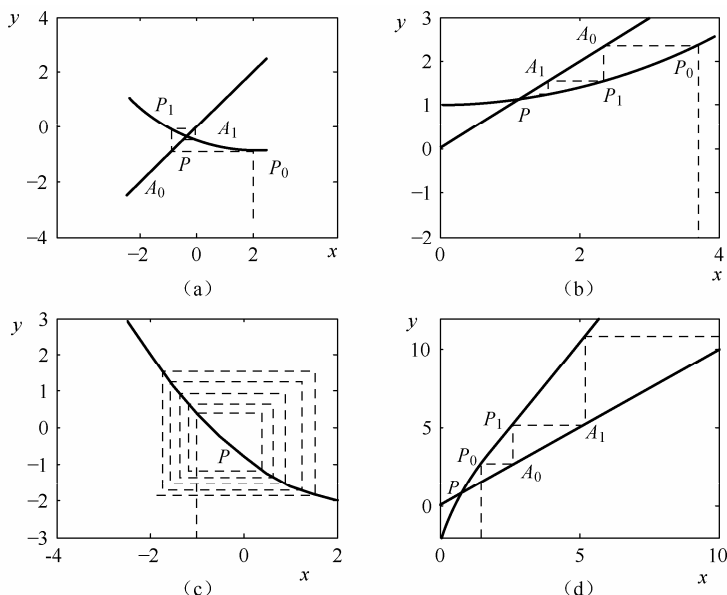


图 3.3.1 迭代法的几何意义

对于  $x^*$  的某个初始近似值  $x_0$ , 在曲线  $x = \varphi(x)$  上可以确定以  $x_0$  为横坐标的一点  $P_0$ ,  $P_0$  的纵坐标为  $x_1 = \varphi(x_0)$ , 过点  $P_0$  作  $x$  轴的平行线交直线  $y = x$  于  $A_0$ , 过  $A_0$  作  $y$  轴的平行线交曲线  $y = \varphi(x)$  于  $P_1$ , 则  $P_1$  的横坐标为  $x_2 = \varphi(x_1)$ , 如此继续下去, 在曲线  $y = \varphi(x)$  上就得到点列  $P_1, P_2, P_3, \dots$ , 其横坐标  $x_1, x_2, x_3, \dots$  由迭代公式  $x_{k+1} = \varphi(x_k)$  求得. 如果点列  $P_1, P_2, P_3, \dots$  越来越逼近交点  $P$ , 则迭代法收敛, 否则迭代法发散. 从图 3.3.1 可以看出, 图 3.3.1 (a) 和 3.3.1 (b) 两种情形是收敛的, 其共同特点是曲线  $y = \varphi(x)$  走势很缓, 即  $|\varphi'(x)| < 1$ ; 而图 3.3.1 (c) 和 3.3.1 (d) 两种情形是发散的, 其共同特点是曲线  $x = \varphi(x)$  走势很陡, 即  $|\varphi'(x)| \geq 1$ .

下面给出迭代法式 (3.3.3) 的收敛性基本定理.

**定理 3.3.1 (收敛性基本定理)** 设迭代函数  $\varphi(x)$  满足如下条件:

- ①  $\varphi(x)$  在  $[a, b]$  内连续, 在  $(a, b)$  内可导;
- ② 映内性, 对任意的  $x \in [a, b]$ , 有  $\varphi(x) \in [a, b]$ ;
- ③ 压缩性, 存在一个常数  $L (0 < L < 1)$  使得在  $[a, b]$  内,  $|\varphi'(x)| \leq L < 1$ .

则

- ① 函数  $\varphi(x)$  在  $[a, b]$  内存在唯一的不动点  $x^*$ ;
- ② 对于任意的初始值  $x_0 \in [a, b]$ , 由式 (3.3.3) 产生的近似值序列  $\{x_k\}_{k=1}^{\infty} \in [a, b]$ , 并且

$$\lim_{k \rightarrow \infty} x_k = x^*;$$

- ③ 有以下误差不等式



$$|x_k - x^*| \leq \frac{L}{1-L} |x_k - x_{k-1}| \quad (3.3.4)$$

$$|x_k - x^*| \leq \frac{L^k}{1-L} |x_1 - x_0| \quad (3.3.5)$$

**证明** ① 先证  $x = \varphi(x)$  在  $[a, b]$  内存在实根。由于  $\varphi(x)$  在  $[a, b]$  内连续, 设  $\varphi_1(x) = x - \varphi(x)$  在  $[a, b]$  内连续, 并且  $\varphi_1(a) = a - \varphi(a) \leq 0$ ,  $\varphi_1(b) = b - \varphi(b) \geq 0$ , 故存在  $x^* \in [a, b]$ , 使

$$\varphi_1(x^*) = 0$$

即  $x^* = \varphi(x^*)$ 。

再证实根的唯一性。若方程  $x = \varphi(x)$  在  $[a, b]$  内有两个实根  $x^*$  和  $x^{**}$ , 则由微分中值定理及条件  $|\varphi'(x)| \leq L < 1$  可知

$$|x^* - x^{**}| = |\varphi(x^*) - \varphi(x^{**})| = |\varphi'(\xi)| |x^* - x^{**}| \leq L |x^* - x^{**}| \quad (\text{其中 } \xi \text{ 在 } x^* \text{ 和 } x^{**} \text{ 之间})$$

显然, 上式在  $x^* = x^{**}$  时成立, 唯一性得证。

② 当  $x_0 \in [a, b]$  时, 由映内性知  $x_k \in [a, b]$ , 由微分中值定理得

$$x^* - x_{k+1} = \varphi(x^*) - \varphi(x_k) = \varphi'(\xi)(x^* - x_k) \quad (\text{其中 } \xi \text{ 在 } x^* \text{ 和 } x_k \text{ 之间})$$

于是

$$|x^* - x_{k+1}| \leq L |x^* - x_k| \quad (k = 0, 1, 2, \dots)$$

故

$$0 \leq |x^* - x_k| \leq L^k |x^* - x_0|$$

注意,  $0 < L < 1$ , 当  $k \rightarrow \infty$  时,  $L^k \rightarrow 0$ , 从而得

$$\lim_{k \rightarrow \infty} |x^* - x_k| = 0$$

即

$$\lim_{k \rightarrow \infty} x_k = x^*$$

③ 由压缩性和微分中值定理可得

$$\begin{aligned} |x^* - x_{k+1}| &\leq L |x^* - x_k| \quad (k = 0, 1, 2, \dots) \\ |x_{k+1} - x_k| &\leq L |x_k - x_{k-1}| \quad (k = 1, 2, \dots) \end{aligned}$$

从而

$$\begin{aligned} |x_{k+1} - x_k| &= |(x^* - x_k) - (x^* - x_{k+1})| \\ &\geq |x^* - x_k| - |x^* - x_{k+1}| \\ &\geq |x^* - x_k| - L |x^* - x_k| = (1-L) |x^* - x_k| \end{aligned}$$

即

$$|x^* - x_k| \leq \frac{1}{1-L} |x_{k+1} - x_k| \leq \frac{L}{1-L} |x_k - x_{k-1}|$$

同时

$$|x^* - x_k| \leq \frac{1}{1-L} |x_{k+1} - x_k| \leq \frac{L}{1-L} |x_k - x_{k-1}| \leq$$

$$\frac{L^2}{1-L}|x_{k-1}-x_{k-2}|\leq\cdots\leq\frac{L^k}{1-L}|x_1-x_0|$$

在例 3.3.1 中, 由于  $\varphi_1(x)=\sqrt[3]{x+1}$  在  $[1,2]$  内导数存在, 且对任意的  $x\in[1,2]$  都有

$$|\varphi_1'(x)|=\left|\frac{1}{3}(x+1)^{-\frac{2}{3}}\right|\leq\frac{1}{3\sqrt[3]{4}}<1$$

$$1<\varphi_1(1)\leq\varphi_1(x)\leq\varphi_1(2)<2$$

满足定理 3.3.1 的各项条件, 所以对  $x_0=1.5\in[1,2]$ , 迭代法

$$x_{k+1}=\sqrt[3]{x_k+1}\quad(k=0,1,2,\cdots)$$

必收敛于方程在  $[1,2]$  内的唯一实根  $x^*$ 。

显然  $\varphi_2(x)$  不满足定理 3.3.1 的条件, 故  $x_{k+1}=\varphi_2(x_k)$  必发散。另外, 从误差估计式 (3.3.5) 可知, 常数  $L$  越小, 收敛速度越快, 可以用来估计迭代次数  $k$ 。在例 3.3.1 中, 若要求近似根  $x_k$  的误差不超过  $10^{-5}$ , 则由该误差估计式可知, 只要使  $k$  满足  $\frac{L^k}{1-L}|x_1-x_0|\leq 10^{-5}$ , 将

$x_0=1.5, x_1=1.35721, L=\frac{1}{3\sqrt[3]{4}}\approx 0.21$  代入, 可得  $k\geq 6.3$ , 故迭代 7 次即可。

式 (3.3.4) 表明, 要使  $|x^*-x_k|\leq\varepsilon$ , 只需  $|x_{k-1}-x_k|\leq\varepsilon$  即可。因此, 可以用迭代前后的两次近似根的差的绝对值大小, 来判断  $x_k$  是否满足精度, 从而可以作为终止迭代的条件。

**例 3.3.2** 能否用迭代法求解下列方程? 如果不能, 试将方程改写成能用迭代法求解的形式:

$$(1) \ x=(\cos x+\sin x)/4; \quad (2) \ x=4-2^x$$

**分析** 判断方程  $x=\varphi(x)$  能不能用迭代法求根, 最关键的是  $\varphi(x)$  在根的邻域能否满足条件  $|\varphi'(x)|\leq L<1$ 。

**解** (1)  $\varphi(x)=(\cos x+\sin x)/4$  对任意的  $x\in(-\infty,+\infty)$ , 恒有

$$|\varphi'(x)|=|(\cos x-\sin x)/4|\leq 1/2<1$$

故能用迭代法求解方程的近似根。

(2) 对方程  $x-4+2^x=0$ , 设  $f(x)=x-4+2^x$ , 则  $f(1)<0, f(2)>0$ , 故  $[1,2]$  为方程的隔离区间。题中  $\varphi(x)=4-2^x, |\varphi'(x)|=|-2^x\ln 2|>2\ln 2\approx 1.36829>1$ , 不满足迭代法收敛性条件, 故不能用迭代法进行求解。若把原方程改为  $x=\ln(4-x)/\ln 2$ , 此时,  $\varphi(x)=\ln(4-x)/\ln 2, |\varphi'(x)|=\left|\frac{-1}{4-x}\times\frac{1}{\ln 2}\right|<\frac{1}{4-2}\times\frac{1}{\ln 2}=\frac{1}{2\ln 2}<1$ , 则可用迭代公式  $x_{k+1}=\ln(4-x_k)/\ln 2$  求该方程的近似根。

**例 3.3.3** 用适当的迭代公式证明:  $\lim_{k\rightarrow\infty}\underbrace{\sqrt{2+\sqrt{2+\cdots+\sqrt{2}}}}_{k\uparrow}=2$ 。

**证明** 考虑迭代公式  $\begin{cases} x_0=0 \\ x_{k+1}=\sqrt{2+x_k}, k=0,1,2,\cdots \end{cases}$

则  $x_1=\sqrt{2}, x_2=\sqrt{2+\sqrt{2}}, x_k=\sqrt{2+\underbrace{\sqrt{2+\cdots+\sqrt{2}}}_{k\uparrow}}$ , 记  $\varphi(x)=\sqrt{2+x}$ , 则  $\varphi'(x)=\frac{1}{2\sqrt{2+x}}$ 。当  $x\in[0,2]$

时,  $\varphi(x)\in[\varphi(0),\varphi(2)]\in[0,2]$ ,  $|\varphi'(x)|\leq\varphi'(0)=\frac{1}{2\sqrt{2}}<1$ , 因而, 所讨论的迭代公式产生的序列

$\{x_k\}_{k=0}^{\infty}$  收敛于方程  $x = \sqrt{2+x}$  在  $[0, 2]$  内的唯一根  $x^* = 2$ ，即  $\lim_{k \rightarrow \infty} x_k = \lim_{k \rightarrow \infty} \sqrt{2 + \sqrt{2 + \cdots + \sqrt{2}}} = 2$ 。

如图 3.3.2 所示为迭代法的算法框图，算法中  $x_0$  为初值， $\varepsilon$  为精度， $N$  为最大迭代次数， $\varphi(x)$  为迭代函数。

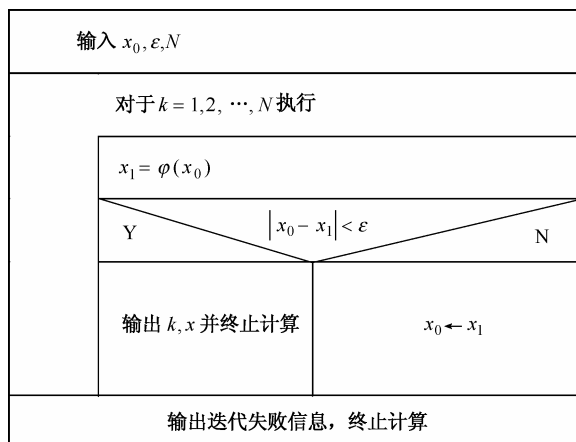


图 3.3.2 迭代法的算法框图

在方程求根的迭代法中，迭代函数  $\varphi(x)$  的确定至关重要，它直接影响着迭代法的收敛性。但在实际应用中，同一个方程可以等价导出不同的迭代函数，而且要严格地利用定理 3.3.1 的条件来判断迭代公式在整个区间  $[a, b]$  内收敛（全局收敛）也非常困难，因此常常判断迭代公式的局部收敛性。

**定义 3.3.1（局部收敛）** 设  $x^*$  是迭代函数  $\varphi(x)$  的不动点，若存在  $x^*$  的某个邻域  $N(x^*, \delta) : |x - x^*| \leq \delta$ ，使得对任意初值  $x_0 \in N(x^*, \delta)$ ，由迭代公式（3.3.3）生成的序列  $\{x_k\}_{k=0}^{\infty} \subset N(x^*, \delta)$ ，且有  $\lim_{k \rightarrow \infty} x_k = x^*$ ，则称迭代公式（3.3.3）局部收敛。

**定理 3.3.2（局部收敛性定理）** 设  $x^*$  是迭代函数  $\varphi(x)$  不动点，若  $\varphi'(x)$  在  $x^*$  的某个邻域内连续，并且有  $|\varphi'(x^*)| < 1$ ，则称迭代公式（3.3.3）局部收敛。

**证明** 由于  $\varphi'(x)$  在  $x^*$  的某个邻域内连续，且  $|\varphi'(x^*)| < 1$ ，则必存在  $x^*$  的一个邻域  $N(x^*, \delta)$  和常数  $L (0 \leq L < 1)$  使得对  $\forall x \in N(x^*, \delta)$ ，有  $|\varphi'(x)| \leq L$ 。

由微分中值定理可知，对任何  $x \in N(x^*, \delta)$ ，都有

$$|\varphi(x) - x^*| = |\varphi(x) - \varphi(x^*)| = |\varphi'(\xi)| |x - x^*| \leq L |x - x^*| < \delta \quad (\text{其中, } \xi \text{ 在 } x \text{ 与 } x^* \text{ 之间})$$

这说明  $\varphi(x) \in N(x^*, \delta)$

于是由定理 3.3.1 和定义 3.3.1 可知，迭代公式（3.3.3）局部收敛。

**例 3.3.4** 为求方程  $x^3 - x^2 - 1 = 0$  在  $x_0 = 1.5$  附近的一个根，将方程改写为下列 5 种等价形式，并建立相应的迭代公式。

(1) $x = 1 + \frac{1}{x^2}$	迭代公式	$x_{k+1} = 1 + \frac{1}{x_k^2}$
(2) $x = \sqrt[3]{1 + x^2}$	迭代公式	$x_{k+1} = \sqrt[3]{1 + x_k^2}$
(3) $x = \frac{1}{\sqrt{x-1}}$	迭代公式	$x_{k+1} = \frac{1}{\sqrt{x_k - 1}}$

$$(4) \quad x = \sqrt{x^3 - 1} \quad \text{迭代公式} \quad x_{k+1} = \sqrt{x_k^3 - 1}$$

$$(5) \quad x = \frac{1}{x^2 - x} \quad \text{迭代公式} \quad x_{k+1} = \frac{1}{x_k^2 - x_k}$$

试分析每种迭代公式的收敛性，并取一种公式求出具有 4 位有效数字的近似根。

解 取  $x_0 = 1.5$  的一个邻域  $[1.45, 1.55]$  来分析。

$$(1) \quad \varphi(x) = 1 + \frac{1}{x^2}, \quad |\varphi'(x)| = \left| \frac{-2}{x^3} \right| \leq \left| \frac{2}{1.45^3} \right| = 0.65603 < 1$$

所以迭代公式 (1) 局部收敛。

$$(2) \quad \varphi(x) = \sqrt[3]{1+x^2}, \quad |\varphi'(x)| = \frac{2x}{3(1+x^2)^{2/3}} \leq \frac{2 \times 1.55}{[3(1+1.45^2)]^{2/3}} = 0.74581 < 1$$

所以迭代公式 (2) 局部收敛。

$$(3) \quad \varphi(x) = \frac{1}{\sqrt{x-1}}, \quad |\varphi'(x)| = \left| \frac{-1}{2(x-1)^{3/2}} \right| \geq \frac{1}{2(1.55-1)^{3/2}} = 1.22581 > 1$$

所以迭代公式 (3) 发散。

$$(4) \quad \varphi(x) = \sqrt{x^3 - 1}, \quad |\varphi'(x)| = \left| \frac{3x^2}{2(x^3 - 1)^{1/2}} \right| \geq \left| \frac{3 \times 1.45^2}{2 \times (1.55^3 - 1)^{1/2}} \right| = 1.91088 > 1$$

所以迭代公式 (4) 发散。

$$(5) \quad \varphi(x) = \frac{1}{x^2 - x}, \quad |\varphi'(x)| = \left| \frac{2x-1}{(x^2 - x)^2} \right| \geq \left| \frac{2 \times 1.45 - 1}{(1.55^2 - 1.55)^2} \right| = 3.9903396 > 1$$

所以迭代公式 (5) 发散。

取迭代公式 (2)，计算结果见表 3.3.2。

表 3.3.2 例 3.3.4 迭代公式 (2) 的计算结果

$k$	$x_k$	$k$	$x_k$	$k$	$x_k$
0	1.5	5	1.466 243 01	10	1.465 583
1	1.481 248 03	6	1.465 876 82	11	1.465 577
2	1.472 705 73	7	1.465 710 24	12	1.465 574
3	1.468 817 31	8	1.465 634 46	13	1.465 572
4	1.467 047 97	9	1.465 599 99	14	1.465 572

因此，取  $x^* \approx x_{14} \approx 1.465572$ 。

当迭代公式收敛时，收敛速度的快慢可用收敛阶来衡量。

**定义 3.3.2 (收敛阶)** 设序列  $\{x_k\}_{k=0}^{\infty}$  收敛到  $x^*$ ，并记误差  $e_k = |x_k - x^*|$ ，若存在常数  $p \geq 1$  和  $c \neq 0$ ，使得

$$\lim_{k \rightarrow \infty} \frac{e_{k+1}}{e_k^p} = c \quad (3.3.6)$$

则称序列  $\{x_k\}_{k=0}^{\infty}$  是  $p$  阶收敛的。当  $p=1$  时，称为线性收敛；当  $p>1$  时，称为超线性收敛；当  $p=2$

时, 称为二次收敛或平方收敛。

由式 (3.3.6) 可知, 当  $k \rightarrow \infty$  时,  $e_{k+1}$  是  $e_k$  的  $p$  阶无穷小量, 因此, 阶数  $p$  越大, 收敛就越快。当然, 线性收敛时, 必有  $0 < |c| \leq 1$ 。

**例 3.3.5** 设  $a > 0, x_0 > 0$ , 证明迭代公式  $x_{k+1} = x_k(x_k^2 + 3a)/(3x_k^2 + a)$  是计算  $\sqrt{a}$  的 3 阶方法, 并求  $\lim_{k \rightarrow \infty} (\sqrt{a} - x_{k+1})/(\sqrt{a} - x_k)^3$ 。

**证明** 显然, 当  $a > 0, x_0 > 0$  时,  $x_k > 0 (k = 1, 2, \dots)$ , 令  $\varphi(x) = x(x^2 + 3a)/(3x^2 + a)$ , 则

$$\varphi'(x) = \frac{(3x^2 + 3a)(3x^2 + a) - x(x^2 + 3a)6x}{(3x^2 + a)^2} = \frac{3(x^2 - a)^2}{(3x^2 + a)^2}$$

对  $\forall x > 0, |\varphi'(x)| < 1$ , 即迭代收敛。设  $\{x_k\}$  的极限为  $x^*$ , 则有  $x^* = x^*(x^{*2} + 3a)/(3x^{*2} + a)$ , 解得  $x^* = 0, x^* = \pm\sqrt{a}$ 。取  $x^* = \sqrt{a}$ ,  $\lim_{k \rightarrow \infty} x_k = \sqrt{a}$ , 下面只要求

$$\begin{aligned} \lim_{k \rightarrow \infty} \frac{\sqrt{a} - x_{k+1}}{(\sqrt{a} - x_k)^3} &= \lim_{k \rightarrow \infty} \frac{\sqrt{a} - (x_k^3 + 3ax_k)/(3x_k^2 + a)}{(\sqrt{a} - x_k)^3} \\ &= \lim_{k \rightarrow \infty} \frac{(\sqrt{a} - x_k)^3}{(\sqrt{a} - x_k)^3 (3x_k^2 + a)} = \lim_{k \rightarrow \infty} \frac{1}{3x_k^2 + a} = \frac{1}{4a} \end{aligned}$$

故迭代收敛是 3 阶收敛的。

**定理 3.3.3 (整数阶超线性收敛定理)** 设  $x^*$  是迭代函数  $\varphi(x)$  的不动点, 若有正整数  $p \geq 2$ , 使得  $\varphi^{(p)}(x)$  在  $x^*$  的邻域连续, 并且满足

$$\varphi^{(n)}(x^*) = 0, \quad n = 1, 2, \dots, p-1, \quad \text{但 } \varphi^{(p)}(x^*) \neq 0$$

则迭代公式 (3.3.3) 局部收敛, 并且是  $p$  阶收敛的。

**证明** 由  $\varphi'(x^*) = 0$  和定理 3.3.2 可知, 迭代公式 (3.3.3) 局部收敛。将  $\varphi(x_k)$  在  $x^*$  处进行泰勒展开, 得

$$x_{k+1} = \varphi(x_k) = \varphi(x^*) + \varphi'(x^*)(x_k - x^*) + \dots + \frac{\varphi^{(p-1)}(x^*)}{(p-1)!}(x_k - x^*)^{p-1} + \frac{\varphi^{(p)}(\xi_k)}{p!}(x_k - x^*)^p$$

式中,  $\xi_k$  在  $x_k$  与  $x^*$  之间。

由于  $\varphi(x^*) = x^*, \varphi^{(n)}(x^*) = 0, n = 1, 2, \dots, p-1$ , 因此

$$x_{k+1} = x^* + \frac{\varphi^{(p)}(\xi_k)}{p!}(x_k - x^*)^p$$

即

$$\frac{e_{k+1}}{e_k^p} = \frac{x_{k+1} - x^*}{(x_k - x^*)^p} = \frac{\varphi^{(p)}(\xi_k)}{p!}$$

又因为迭代公式收敛, 当  $k \rightarrow \infty$  时,  $x_k \rightarrow x^*$ , 从而  $\xi_k \rightarrow x^*$ , 故有  $\frac{e_{k+1}}{e_k^p} = \frac{\varphi^{(p)}(x^*)}{p!} \neq 0$ 。故迭代公式是  $p$  阶收敛的。

**例 3.3.6** 试确定常数  $p, q, r$ , 使迭代公式  $x_{k+1} = px_k + qa/x_k^2 + ra^2/x_k^5$  产生的序列  $\{x_k\}$  收敛到  $\sqrt[3]{a}$ , 并使其收敛阶尽可能高。

**解** 已知迭代函数  $\varphi(x) = px + qa/x^2 + ra^2/x^5$ , 根据迭代收敛阶次的定理 3.3.3, 要想使所

研究的迭代公式具有尽可能高的收敛阶次, 迭代函数  $\varphi(x)$  应首先满足条件:  $x^* = \varphi(x^*)$ ,  $\varphi'(x^*) = 0$ ,  $\varphi''(x^*) = 0$ , 由此可确定  $p, q, r$  应满足的方程。

$$\begin{aligned} \text{由 } x^* = \varphi(x^*), \text{ 得 } \sqrt[3]{a} &= \sqrt[3]{a}p + qa/\sqrt[3]{a^2} + ra^2/\sqrt[3]{a^5}, \text{ 即} \\ p + q + r &= 1 \end{aligned} \quad (3.3.7)$$

$$\begin{aligned} \text{由 } \varphi'(x^*) = 0, \text{ 得 } \varphi'(\sqrt[3]{a}) &= p - 2qa/(\sqrt[3]{a})^3 - 5ra^2/(\sqrt[3]{a})^6 = 0, \text{ 即} \\ p - 2q - 5r &= 0 \end{aligned} \quad (3.3.8)$$

$$\begin{aligned} \text{由 } \varphi''(x^*) = 0, \text{ 得 } \varphi''(\sqrt[3]{a}) &= 6qa/(\sqrt[3]{a})^4 + 30ra^2/(\sqrt[3]{a})^7 = 0, \text{ 即} \\ q + 5r &= 0 \end{aligned} \quad (3.3.9)$$

联立式 (3.3.7)、式 (3.3.8)、式 (3.3.9), 得  $p = q = 5/9, r = -1/9$ , 而且由于  $\varphi'''(\sqrt[3]{a}) \neq 0$ , 故所确定的迭代公式  $x_{k+1} = 5/9x_k + 5/9a/x_k^2 - 1/9a^2/x_k^5$ , 其收敛阶次为 3 阶。

### 3.3.3 迭代法的改善

对于收敛的迭代法, 按照精度要求, 经过若干次的迭代, 总能求出方程的近似值。但在实际应用中, 如果迭代公式的收敛速度太慢, 必然影响计算效率, 从而失去实用价值。因此, 有必要对迭代过程进行改善。

由定理 3.3.2 可知, 若  $x^*$  是  $\varphi(x)$  的不动点,  $\varphi'(x)$  在  $x^*$  的某个领域  $N(x^*, \delta)$  内连续, 且  $0 < |\varphi'(x^*)| < 1$ , 则迭代公式 (3.3.3) 局部线性收敛。当  $|\varphi'(x^*)| > 1$  时, 肯定不收敛。下面介绍当  $\varphi(x^*) \neq 1$  时的改善收敛方法——Steffensen 迭代法。

按照收敛阶的定义 3.3.2, 若迭代公式 (3.3.3) 线性收敛, 记  $e_k = x_k - x^*$ , 则有

$$\lim_{k \rightarrow \infty} \frac{e_{k+1}}{e_k} = \lim_{k \rightarrow \infty} \frac{x_{k+1} - x^*}{x_k - x^*} = c \neq 0$$

因此, 当  $k$  充分大时, 比值  $\frac{e_{k+1}}{e_k}$  变化不大, 即有  $\frac{x_{k+1} - x^*}{x_k - x^*} \approx \frac{x_{k+2} - x^*}{x_{k+1} - x^*}$ 。

从上式中解出  $x^*$ , 得到比  $x_k, x_{k+1}$  更好的一个近似值

$$x^* \approx \frac{x_{k+2}x_k - x_{k+1}^2}{x_{k+2} - 2x_{k+1} + x_k} = x_k - \frac{(x_{k+1} - x_k)^2}{x_{k+2} - 2x_{k+1} + x_k}$$

令  $y_k = \varphi(x_k)$ ,  $z_k = \varphi(y_k)$ ,  $x^*$  新的近似值记为  $x_{k+1}$ , 则有

$$x_{k+1} = \frac{x_k z_k - y_k^2}{z_k - 2y_k + x_k} = x_k - \frac{(y_k - x_k)^2}{z_k - 2y_k + x_k} \quad (k = 0, 1, 2, \dots)$$

它的不动点迭代形式

$$x_{k+1} = \psi(x_k) \quad (k = 0, 1, 2, \dots) \quad (3.3.10)$$

迭代函数

$$\psi(x) = \frac{x\varphi(\varphi(x)) - [\varphi(x)]^2}{\varphi(\varphi(x)) - 2\varphi(x) + x} = x - \frac{[\varphi(x) - x]^2}{\varphi(\varphi(x)) - 2\varphi(x) + x} \quad (3.3.11)$$

迭代公式 (3.3.10) 和式 (3.3.11), 称为 Steffensen 迭代法。

**定理 3.3.4 (Steffensen 迭代法的收敛定理)** 设函数  $\psi(x)$  为由  $\varphi(x)$  按式 (3.3.11) 定义

的 Steffensen 迭代法的迭代函数:

① 若  $x^*$  是  $\varphi(x)$  的不动点,  $\varphi'(x)$  在  $x^*$  处连续, 且  $\varphi'(x^*) \neq 1$ , 则  $x^*$  也是  $\psi(x)$  的不动点; 反之, 若  $x^*$  是  $\psi(x)$  的不动点, 则  $x^*$  也是  $\varphi(x)$  的不动点。

② 若  $x^*$  是  $\varphi(x)$  的不动点,  $\varphi'(x)$  在  $x^*$  处连续, 且  $\varphi'(x^*) \neq 1$ , 则 Steffensen 迭代公式 (3.3.10), 至少是平方收敛的。

**证明** ① 若  $x^*$  是  $\varphi(x)$  的不动点, 即  $x^* = \varphi(x^*)$ , 则当  $x = x^*$  时, 式 (3.3.11) 的分子和分母都为 0, 此时对式 (3.3.11) 的极限使用罗必塔 (L'Hospital) 法则, 注意  $\varphi'(x^*) \neq 1$ , 得

$$\begin{aligned}\lim_{x \rightarrow x^*} \psi(x) &= \lim_{x \rightarrow x^*} \frac{\varphi(\varphi(x)) + x\varphi'(\varphi(x))\varphi'(x) - 2\varphi(x)\varphi'(x)}{\varphi'(\varphi(x))\varphi'(x) - 2\varphi'(x) + 1} \\ &= \frac{x^* [\varphi'(x^*) - 1]^2}{[\varphi'(x^*) - 1]^2} = x^*\end{aligned}$$

从而  $x^* = \psi(x^*)$ , 即  $x^*$  为  $\psi(x)$  的不动点。反之, 若  $x^* = \psi(x^*)$ , 则由式 (3.3.11) 可得  $x^* = \varphi(x^*)$ 。

② 由本定理的条件和结论①可知,  $x^*$  是  $\psi(x)$  的不动点, 于是, 由定理 3.3.3 可知, 只需证明  $\psi'(x^*) = 0$  即可。

对式 (3.3.11) 两边求导数, 得  $\psi'(x) = 1 - \frac{\omega(x)}{\nu(x)}$ , 即

$$1 - \psi'(x) = \frac{\omega(x)}{\nu(x)} \quad (3.3.12)$$

式中

$$\begin{aligned}\omega(x) &= 2[\varphi(x) - x][\varphi'(x) - 1][\varphi(\varphi(x)) - 2\varphi(x) + x] - [\varphi(x) - x]^2[\varphi'(\varphi(x))\varphi'(x) - 2\varphi'(x) + 1] \\ \nu(x) &= [\varphi(\varphi(x)) - 2\varphi(x) + x]^2\end{aligned}$$

可以算出  $\omega''(x^*) = \nu''(x^*) = 2[\varphi'(x^*) - 1]^4$

由于  $\varphi'(x^*) \neq 1$ , 因此  $\omega''(x^*) = \nu''(x^*) \neq 0$ 。

对式 (3.3.12) 两边求极限, 并对左边两次使用罗必塔法则, 得

$$1 - \psi'(x^*) = \lim_{x \rightarrow x^*} [1 - \psi'(x)] = \lim_{x \rightarrow x^*} \frac{\omega''(x)}{\nu''(x)} = 1$$

从而  $\psi'(x^*) = 0$ 。由此可知, 在  $\varphi'(x^*) \neq 1$  的前提下, Steffensen 迭代法至少是平方收敛的。

**例 3.3.7** 利用例 3.3.1 中的第二种等价形式, 求方程  $f(x) = x^3 - x - 1 = 0$  在  $x = 1.5$  附近的实根。

**解** 在例 3.3.1 中, 第二种等价形式  $x_{k+1} = \varphi(x_k) = x_k^3 - 1$  发散。现在用  $\varphi_2(x) = x^3 - 1$  构造 Steffensen 迭代法

$$y_k = x_k^3 - 1, \quad z_k = y_k^3 - 1, \quad x_{k+1} = x_k - \frac{(y_k - x_k)^2}{z_k - 2y_k + x_k}$$

取  $x_0 = 1.5$  计算得

$$x_1 = 1.416\ 292\ 97, \dots, x_5 = 1.324\ 717\ 99, x_6 = 1.324\ 717\ 96$$

$x_6$  具有 8 位有效数字。

## 3.4 牛顿迭代法

### 3.4.1 牛顿迭代公式及其几何意义

设函数  $f(x)$  连续可导,  $x^*$  是方程  $f(x) = 0$  的实根,  $x_k$  是方程根的某个近似值, 将函数  $f(x)$  在点  $x_k$  进行一阶泰勒展开, 则有

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k)$$

于是, 有

$$0 = f(x^*) \approx f(x_k) + f'(x_k)(x^* - x_k)$$

当  $f'(x_k) \neq 0$  时, 有

$$x^* \approx x_k - \frac{f(x_k)}{f'(x_k)}$$

右端是方程的根  $x^*$  的又一个新的近似值, 记为  $x_{k+1}$ , 便得迭代公式

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad (k = 0, 1, 2, \dots) \quad (3.4.1)$$

式 (3.4.1) 称为牛顿 (Newton) 迭代公式, 相应的迭代函数为  $\varphi(x) = x - \frac{f(x)}{f'(x)}$ 。显然, 迭代

方程  $\varphi(x) = x - \frac{f(x)}{f'(x)}$  与原方程  $f(x) = 0$  等价。

牛顿迭代公式 (3.4.1) 具有明显的几何意义。方程  $f(x) = 0$  的根  $x^*$  在几何上表示曲线  $y = f(x)$  与  $x$  轴交点的横坐标 (见图 3.4.1)。对于  $x^*$  的某个近似值  $x_k$ , 过曲线  $y = f(x)$  上对应的点  $(x_k, f(x_k))$  作  $f(x)$  的切线, 其切线方程为  $y - f(x_k) = f'(x_k)(x - x_k)$ , 求方程与  $x$  轴的交点, 即得  $x^*$  的新近似值  $x_{k+1}$ , 它满足  $0 - f(x_k) = f'(x_k)(x_{k+1} - x_k)$ 。即  $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$ , 这正是牛顿迭代公式的计算结果。因此牛顿迭代法又称为切线法。

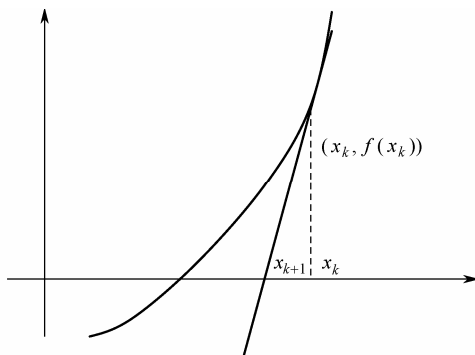


图 3.4.1 牛顿迭代公式的几何意义



### 3.4.2 牛顿迭代公式的收敛性

**定理 3.4.1** 设  $x^*$  是方程  $f(x)=0$  的单根, 并且  $f''(x)$  在  $x^*$  的邻域上连续, 则牛顿迭代公式 (3.4.1) 至少平方局部收敛。

**证明** 将式 (3.4.1) 写成不动点迭代形式  $x_{k+1} = \varphi(x_k)$ , 式中  $\varphi(x) = x - \frac{f(x)}{f'(x)}$ , 称其为牛顿迭代函数。

因为  $x^*$  为  $f(x)=0$  的单根, 所以  $f'(x^*) \neq 0$ , 从而  $x^* = \varphi(x^*)$ , 即  $x^*$  是  $\varphi(x)$  的不动点。对迭代函数  $\varphi(x)$  求导数, 得

$$\varphi'(x) = 1 - \frac{[f'(x)]^2 - f(x)f''(x)}{[f'(x)]^2} = \frac{f(x)f''(x)}{[f'(x)]^2}$$

因为  $f'(x^*) \neq 0$ , 所以  $\varphi'(x^*) = 0$ , 根据定理 3.3.2 可知牛顿迭代公式 (3.4.1) 局部收敛。将  $f(x^*)$  在  $x_k$  处进行泰勒展开, 得

$$0 = f(x^*) = f(x_k) + f'(x_k)(x^* - x_k) + \frac{f''(\xi_k)}{2}(x^* - x_k)^2 \quad (\text{其中 } \xi_k \text{ 在 } x_k \text{ 与 } x^* \text{ 之间})$$

将式 (3.4.1) 改写为  $f(x_k) - f'(x_k)x_k = -f'(x_k)x_{k+1}$ , 代入上式得

$$0 = f'(x_k)(x^* - x_{k+1}) + \frac{f''(\xi_k)}{2}(x^* - x_k)^2$$

即

$$\frac{e_{k+1}}{e_k^2} = \frac{f''(\xi_k)}{2f'(x_k)} \quad (e_k = |x_k - x^*|)$$

令  $k \rightarrow \infty$ , 由局部收敛性可知  $x_k \rightarrow x^*$ , 从而  $\xi_k \rightarrow x^*$ , 所以有

$$\lim_{k \rightarrow \infty} \frac{e_{k+1}}{e_k^2} = \frac{f''(x^*)}{2f'(x^*)} = c \quad (c \neq 0)$$

因此牛顿迭代公式至少平方局部收敛。

**例 3.4.1** 用牛顿迭代公式求方程  $f(x) = x^3 - x - 1 = 0$  在  $x = 1.5$  附近的根。

**解** 取  $x_0 = 1.5$

$$\begin{aligned} x_{k+1} &= x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k^3 - x_k - 1}{3x_k^2 - 1} \\ x_1 &= x_0 - \frac{x_0^3 - x_0 - 1}{3x_0^2 - 1} = 1.5 - \frac{1.5^3 - 1.5 - 1}{3 \times (1.5)^2 - 1} \approx 1.34783 \\ x_2 &= x_1 - \frac{x_1^3 - x_1 - 1}{3x_1^2 - 1} \approx 1.32520 \\ x_3 &= x_2 - \frac{x_2^3 - x_2 - 1}{3x_2^2 - 1} \approx 1.32472 \\ x_4 &= x_3 - \frac{x_3^3 - x_3 - 1}{3x_3^2 - 1} \approx 1.32472 \end{aligned}$$

与例 3.3.1 比较, 牛顿迭代法迭代 3 次就能达到 6 位有效数字, 收敛速度明显加快。不过, 如果选取初值  $x_0 = 0.6$ , 按牛顿迭代公式 (3.4.1) 计算得到第一步迭代值为  $x_1 = 17.9$ , 它比  $x_0$  更远

离方程的根  $x^*$ ，因此迭代过程发散。这说明牛顿迭代公式是局部收敛的，其收敛性与初值  $x_0$  的选取有关。

下面介绍一种在方程  $f(x)=0$  的有根区间  $[a,b]$  内选取初值的简单方法。

由于曲线  $y=f(x)$  在根  $x^*$  附近有极值点或拐点时，牛顿迭代公式有可能不收敛，因此总假设在  $[a,b]$  内  $f'(x) \neq 0$ ，且  $f''(x)$  在  $[a,b]$  内不变号。

根据  $f(x)$  和  $f''(x)$  的正负，曲线  $y=f(x)$  只可能有如图 3.4.2 所示的 4 种情况。

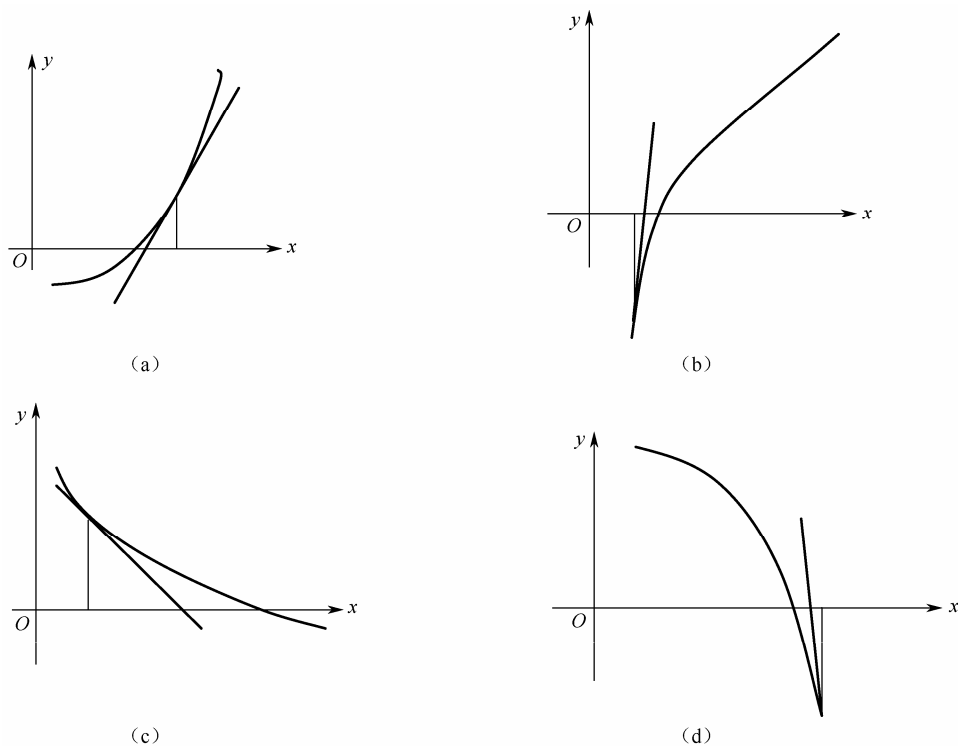


图 3.4.2 曲线  $y=f(x)$  的 4 种情况

由图 3.4.2 不难看出，只要把  $x_0$  选取得使  $f(x_0)$  和  $f''(x_0)$  同号，即  $f(x_0) \times f''(x_0) > 0$ ，则迭代过程必收敛。

例 3.4.1 中，有  $f(0.6) = 0.6^3 - 0.6 - 1 = -1.384 < 0$ ， $f''(0.6) = 6 \times 0.6 = 3.6 > 0$ ， $f(0.6) \times f''(0.6) < 0$ ，所以选取  $x_0 = 0.6$  时，迭代过程发散。而  $f(1.5) = 1.5^3 - 1.5 - 1 = 0.875 > 0$ ， $f''(1.5) = 6 \times 1.5 = 9 > 0$ ， $f(1.5) \times f''(1.5) > 0$ ，所以选取  $x_0 = 1.5$  时，迭代过程收敛。

**例 3.4.2** 利用牛顿迭代公式求  $\sqrt{115}$  的近似值。

**解** 方法 1：设  $f(x) = x^2 - 115$ ，则求  $\sqrt{115}$  的值变成求  $f(x) = 0$  的正根。由于  $f(10) = -15 < 0$ ， $f(11) = 6 > 0$ ，因此方程  $f(x) = 0$  在  $(10, 11)$  内有一根。取  $x_0 = 11$ ，由牛顿迭代公式

$$x_{k+1} = x_k - \frac{(x_k^2 - 115)}{2x_k} \quad (k = 0, 1, 2, \dots)$$

得  $x_1 = 10.72727273$ ， $x_2 = 10.72380586$ ， $x_3 = 10.72380530$

此时， $x_2, x_3$  已有 8 位数相同，取  $x^* \approx x_3 = 10.723805$ 。

方法 2: 对方程  $f(x) = 1 - \frac{a}{x^2} = 0$  应用牛顿迭代公式, 求  $\sqrt{115}$  的值。

因为  $f(x) = 1 - \frac{a}{x^2}$ ,  $f'(x) = \frac{2a}{x^3}$ ,  $x \neq 0$ , 所以牛顿迭代公式为

$$x_{k+1} = x_k - \frac{1 - \frac{a}{x_k^2}}{\frac{2a}{x_k^3}} = \frac{1}{2} x_k \left( 3 - \frac{x_k^2}{a} \right) \quad (k = 0, 1, 2, \dots)$$

易知  $f''(x) = -\frac{6a}{x^4} < 0$ , 故取  $x_0 \in (0, \sqrt{a})$  时, 迭代收敛。

对于  $\sqrt{115}$ , 取  $x_0 = 9$ , 得

$x_1 = 10.33043478$ ,  $x_2 = 10.70242553$ ,  $x_3 = 10.7237414$ ,  $x_4 = 10.72380529$ ,  $x_5 = 10.72380529$  故  $\sqrt{115} \approx 10.72380529$ 。

以上讨论的是局部收敛性, 对于某些非线性方程, 牛顿迭代公式具有全局收敛性。

**例 3.4.3** 设有常数  $c > 0$ , 用牛顿迭代公式求方程  $x^2 - c = 0$  的根  $\sqrt{c}$ , 试证: 任取初值  $x_0 > 0$ , 迭代公式都收敛到  $\sqrt{c}$ 。

**证明** 因为  $f(x) = x^2 - c$ ,  $f'(x) = 2x$ , 所以牛顿迭代公式为

$$x_{k+1} = x_k - \frac{(x_k^2 - c)}{2x_k} = \frac{1}{2} \left( x_k + \frac{c}{x_k} \right) \quad (3.4.2)$$

又因为

$$x_{k+1} - \sqrt{c} = \frac{1}{2} \left( x_k + \frac{c}{x_k} \right) - \sqrt{c} = \frac{1}{2x_k} (x_k^2 - 2x_k\sqrt{c} + c) = \frac{1}{2x_k} (x_k - \sqrt{c})^2 \geq 0$$

即对于任意的初值  $x_0 > 0$ , 有

$$x_k \geq \sqrt{c} \quad (k = 1, 2, \dots)$$

从而

$$x_k - x_{k+1} = \frac{1}{2x_k} (x_k^2 - c) \geq 0 \quad (k = 1, 2, \dots)$$

由此可得, 迭代序列  $\{x_k\}_{k=1}^{\infty}$  是有下界的单调非增序列, 从而有极限  $x^*$ 。对式 (3.4.2) 两边取极限, 得

$$x^* = \lim_{k \rightarrow \infty} x_{k+1} = \lim_{k \rightarrow \infty} \frac{1}{2} \left( x_k + \frac{c}{x_k} \right) = \frac{1}{2} \left( x^* + \frac{c}{x^*} \right)$$

所以

$$x^* = \sqrt{c}$$

**例 3.4.4** 应用牛顿迭代公式于方程  $x^3 - a = 0$ , 导出求立方根  $\sqrt[3]{a}$  的迭代公式, 并讨论其收敛性。

**解** 方程  $x^3 - a = 0$  的根为  $x^* = \sqrt[3]{a}$ , 用牛顿迭代法, 有

$$x_{k+1} = x_k - \frac{x_k^3 - a}{3x_k^2} = \frac{2x_k}{3} + \frac{a}{3x_k^2} \quad (k = 0, 1, 2, \dots)$$

迭代函数为  $\varphi(x) = \frac{2}{3}x + \frac{a}{3x^2}$ , 则  $\varphi'(x) = \frac{2}{3} - \frac{2a}{3x^3}$ ,  $\varphi'(x^*) = 0$ ,  $\varphi''(x^*) = \frac{2}{\sqrt[3]{a}} \neq 0$ , 故迭代公式二阶

收敛。

还可证明迭代公式全局收敛性。设  $a > 0$ ，对任意  $x_0 > 0$ ，有

$$x_1 = 2x_0/3 + a/(3x_0^2) = (2x_0^3 + a)/(3x_0^2) \geq \sqrt[3]{a}$$

一般地，当  $x_k > 0$  时，有

$$x_{k+1} = 2x_k/3 + a/(3x_k^2) \geq \sqrt[3]{a} \quad (k=0,1,2,\dots)$$

这是因为：  $(x_k - \sqrt[3]{a})^2(2x_k + \sqrt[3]{a}) = 2x_k^3 + a - 3x_k^2\sqrt[3]{a} \geq 0$ ，当  $x_k > 0$  时成立。从而再由

$$\frac{x_{k+1}}{x_k} = \frac{2}{3} + \frac{a}{3x_k^3}$$

可得  $x_{k+1}/x_k \leq 1$ ，即  $x_{k+1} \leq x_k$ 。表明序列  $\{x_k\}$  单调递减，故对任意  $x_0 > 0$ ，迭代序列  $\{x_k\}$  收敛于  $\sqrt[3]{a}$ 。

如图 3.4.3 所示为牛顿迭代法的算法框图，算法中  $x_0, x_1$  分别表示每次迭代的初值和终值， $\varepsilon$  为精度， $N$  为最大迭代次数。

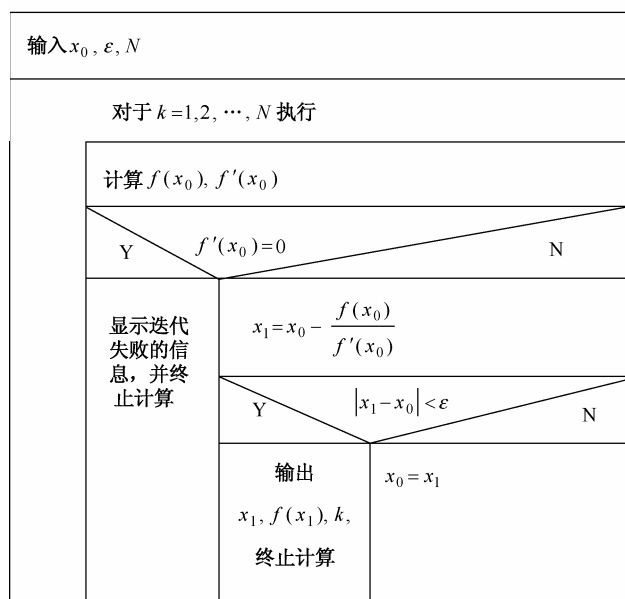


图 3.4.3 牛顿迭代法的算法框图

### 3.4.3 重根情形

由定理 3.4.1 可知，若  $x^*$  是方程  $f(x) = 0$  的单根，则牛顿迭代公式至少具有平方收敛速度。但是，当  $x^*$  为  $f(x) = 0$  的重根时，牛顿迭代法的收敛速度明显地下降。

一般地，设  $x^*$  是  $f(x) = 0$  的  $m$  重根 ( $m > 1$ )，即  $f(x) = (x - x^*)^m g(x)$ ，其中  $g(x)$  有二阶导数，且  $g(x^*) \neq 0$ ，此时，牛顿迭代法的迭代函数为

$$\begin{aligned} \varphi(x) &= x - \frac{f(x)}{f'(x)} = x - \frac{(x - x^*)^m g(x)}{m(x - x^*)^{m-1} g(x) + (x - x^*)^m g'(x)} \\ &= x - \frac{(x - x^*)^{m-1} (x - x^*) g(x)}{(x - x^*)^{m-1} [m g(x) + (x - x^*) g'(x)]} \end{aligned}$$

$$= x - \frac{(x-x^*)g(x)}{mg(x) + (x-x^*)g'(x)}$$

因此

$$\varphi'(x) = \frac{(1-\frac{1}{m}) + (x-x^*)\frac{2g'(x)}{mg(x)} + (x-x^*)^2\frac{g''(x)}{m^2g(x)}}{\left[1 + (x-x^*)\frac{g'(x)}{mg(x)}\right]^2}$$

所以  $\varphi'(x^*) = 1 - \frac{1}{m}$ ，当  $m > 1$  时， $\varphi'(x^*) \neq 0$ ，且有  $|\varphi'(x^*)| < 1$ 。由定理 3.3.3 可知，若  $x^*$  为  $f(x) = 0$  的  $m$  重根，牛顿迭代法是线性收敛的。

为了改善重根时的牛顿法的收敛性，可采用如下方法。

方法 1：如果方程  $f(x) = 0$  的根  $x^*$  的重根数  $m (m \geq 2)$  已知，那么可将迭代函数改写为

$$\varphi_1(x) = x - m \frac{f(x)}{f'(x)}$$

由上面的推导过程不难验证

$$\varphi_1'(x^*) = 0$$

因此，下面的迭代公式至少平方收敛

$$x_{k+1} = \varphi_1(x_k) = x_k - m \frac{f(x_k)}{f'(x_k)} \quad (k = 0, 1, 2, \dots)$$

方法 2：如果方程  $f(x) = 0$  的根  $x^*$  的重根数  $m (m \geq 2)$  未知，那么可将方程  $f(x) = 0$  改写为

$$\varphi_2(x) = \frac{f(x)}{f'(x)} = 0$$

由于  $x^*$  是  $f(x)$  的  $m$  重根， $x^*$  是  $f'(x)$  的  $m-1$  重根，因此  $x^*$  是函数  $\varphi_2(x)$  的单根。对于函数  $\varphi_2(x)$  使用牛顿迭代法，则至少具有平方收敛速度。函数  $\varphi_2(x)$  的牛顿迭代公式为

$$x_{k+1} = x_k - \frac{\varphi_2(x_k)}{\varphi_2'(x_k)} = x_k - \frac{f(x_k)f'(x_k)}{[f'(x_k)]^2 - f(x_k)f''(x_k)} \quad (k = 0, 1, 2, \dots)$$

**例 3.4.5** 用牛顿迭代法及其两种改善方法求方程  $f(x) = x^3 - x^2 - x + 1 = (x+1)(x-1)^2 = 0$  的二重根  $x^* = 1$ 。

**解** 取  $x_0 = 1.5$ ，计算结果见表 3.4.1。

表 3.4.1 例 3.4.5 的计算结果

	$k$	0	1	...	24	25
牛顿法	$x_k$	1.5	1.272 727 273	...	1.000 000 037	1.000 000 019
	$k$	0	1	2	3	4
改善方法 1	$x_k$	1.5	1.045 454 545	1.000 499 500	1.000 000 062	1.000 000 000
改善方法 2	$x_k$	1.5	0.960 784 314	0.999 600 080	0.999 999 960	1.000 000 000

需要指出的是：改善方法 1 的缺点是必须事先知道方程的重根数，而在实际计算中，往往不能事先知道根的重数；改善方法 2 的缺点是每次迭代需要求函数  $f(x)$  的二阶导数，增加了计算量，并且当所求根为单根时，不能改善迭代法的收敛性。

### 3.5 弦截法

牛顿迭代法具有收敛速度快,能求重根等优点,但是每迭代一步,都要计算函数的导数值,计算量很大,尤其是当函数的结构比较复杂或函数不可导时,就很难使用牛顿迭代法。为了克服这些缺点,在实际计算中,常采用函数  $f(x)$  在以  $x_{k-1}, x_k$  为端点的区间内的平均变化率  $\frac{f(x_k)-f(x_{k-1})}{x_k-x_{k-1}}$ , 近似为  $f'(x_k)$ , 此时, 牛顿迭代公式改为

$$x_{k+1} = x_k - \frac{f(x_k)}{f(x_k)-f(x_{k-1})}(x_k - x_{k-1}) \quad (k=1,2,\dots) \quad (3.5.1)$$

设方程  $f(x)=0$  的根  $x^*$  有两个近似值  $x_{k-1}, x_k$ , 如图 3.5.1 所示。

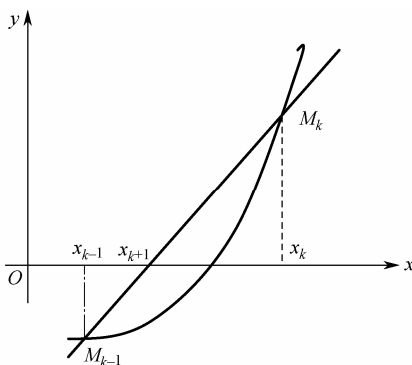


图 3.5.1 弦截法的几何意义

图中, 过  $M_{k-1}(x_{k-1}, f(x_{k-1}))$  和  $M_k(x_k, f(x_k))$  引一条直线, 则该直线与  $x$  轴交点的横坐标  $x$  为

$$x = x_k - \frac{f(x_k)}{f(x_k)-f(x_{k-1})}(x_k - x_{k-1})$$

因此, 迭代公式 (3.5.1) 可以看成截线  $M_{k-1}M_k$  与  $x$  轴交点的横坐标作为  $x^*$  的新的近似值  $x_{k+1}$ , 所以式 (3.5.1) 也称为弦截法, 或称离散牛顿法。与牛顿法相比, 它避免了求函数  $f(x)$  的导数, 但需要两个初始值  $x_0, x_1$ , 且这两个初始值应尽量取在方程  $f(x)=0$  的根  $x^*$  附近。其收敛速度一般比牛顿法慢, 但比线性收敛的方法快。在与牛顿法具有同等的前提条件下, 弦截法式 (3.5.1) 具有局部收敛性, 并且收敛阶  $p = \frac{1+\sqrt{5}}{2} \approx 1.618$ 。由于弦截法是两步法, 它不属于不动点迭代, 因此不能用不动点迭代理论证明它的收敛性。

**例 3.5.1** 用弦截法求方程  $f(x)=x^3-x-1$  在  $[1,1.5]$  的根。

**解** 因为  $f(1)=-1<0$ ,  $f(1.5)=0.875>0$  且  $f(x)$  为单调连续函数, 所以  $f(x)=0$  在  $[1,1.5]$  内有唯一的根, 取  $x_0=1$ ,  $x_1=1.5$ , 用弦截法计算结果见表 3.5.1。

表 3.5.1 例 3.5.1 的计算结果

$k$	$x_k$	$k$	$x_k$
0	1	4	1.325 214
1	1.5	5	1.324 714
2	1.266 667	6	1.324 718
3	1.315 962	...	...

与例 3.3.1 比较, 弦截法收敛速度较快, 但与例 3.4.1 比较, 则收敛速度较慢。

## 本章小结

本章讨论了求解非线性方程的一些数值计算方法。二分法方法简单, 编程容易, 且对函数  $f(x)$  的性质要求不高, 但其收缩速度较慢, 与公比为  $\frac{1}{2}$  的等比级数的收缩速度相同, 因此常被用于求精度不高的近似根或为迭代法提供初值。迭代法是一种逐次逼近的方法, 原理简单, 但存在收敛性和收敛速度的问题。当迭代过程不收敛或只有线性收敛速度时, 可以用 Steffensen 迭代法改善。牛顿法是一种特殊的迭代法, 它除了具有一般迭代法的特点外, 还具有在单根附近的收敛速度为平方收敛的特点, 经过改善可以求方程得重根。但牛顿法对初值得选取要求苛刻, 且需要函数的导数, 因此, 当导数过于复杂时, 可用弦截法求解。弦截法不要求求导数, 但收敛速度较慢, 收敛阶为 1.618, 且需要两个初值。

## 习题 3

3.1 试取  $x_0 = 1$ , 用迭代公式

$$x_{k+1} = \frac{20}{x_k^2 + 2x_k + 10} \quad (k = 0, 1, 2, \dots)$$

求方程  $x^3 + 2x^2 + 10x - 20 = 0$  的根, 要求精确到  $10^{-3}$ 。

3.2 证明方程  $x = \frac{1}{2} \cos x$  有且仅有一个实根, 试确定区间  $[a, b]$ , 使迭代过程  $x_{k+1} = \frac{1}{2} \cos x_k$  对于一切  $x_0 \in [a, b]$  均收敛。

3.3 设方程  $12 - 3x + 2 \cos x = 0$  的迭代公式为  $x_{k+1} = 4 + \frac{2}{3} \cos x_k$ , 回答以下问题:

(1) 证明对  $\forall x_0 \in R$ , 均有  $\lim_{k \rightarrow \infty} x_k = x^*$ , 其中  $x^*$  为方程的根。

(2) 取  $x_0 = 4$ , 求此迭代法的近似根, 使误差不超过  $10^{-3}$ , 并列各次迭代值。

(3) 此迭代法收敛阶是多少? 证明你的结论。

3.4 已知方程  $f(x) = 0$ , 试导出求根的迭代公式

$$x_{k+1} = x_k - \frac{2f'(x_k)f(x_k)}{2[f'(x_k)]^2 - f''(x_k)f(x_k)}$$

并证明当  $f(x) = 0$  的根  $x^*$  为单根时, 公式为 3 阶收敛。

3.5 设  $\varphi(x) = x - p(x)f(x) - q(x)f^2(x)$ , 试确定函数  $p(x)$  和  $q(x)$ , 使求解  $f(x) = 0$  并以  $\varphi$  为迭代公式的迭代法至少 3 阶收敛。

3.6 基于迭代原理证明

$$\sqrt{1 + \sqrt{1 + \sqrt{1 + \dots}}} = \frac{1 + \sqrt{5}}{2}$$

3.7 给定函数  $f(x)$ , 设对一切  $x, f'(x)$  存在且  $0 < m \leq f'(x) \leq M$ , 试证明对于  $0 < \lambda < \frac{2}{M}$  的

任意键值  $\lambda$ ，迭代过程  $x_{k+1} = x_k - \lambda f(x_k)$  均收敛于  $f(x) = 0$  的根  $x^*$ 。

3.8 设函数  $f(x)$  具有二阶连续导数， $f(x^*) = 0$ ， $f'(x^*) \neq 0$ ， $f''(x^*) \neq 0$ ， $\{x_k\}$  是由牛顿迭代法产生的序列，证明  $\lim_{k \rightarrow \infty} \frac{x_{k+1} - x_k}{(x_k - x_{k-1})^2} = -\frac{1}{2} \frac{f''(x^*)}{f'(x^*)}$ 。

3.9 证明用解方程  $(x^2 - a)^2 = 0$  求  $\sqrt{a}$  的牛顿迭代公式

$$x_{k+1} = \frac{3}{4}x_k + \frac{a}{4x_k}$$

仅线性收敛。

3.10 证明用解方程  $(x^3 - a)^2 = 0$  求  $\sqrt[3]{a}$  的牛顿公式

$$x_{k+1} = \frac{5}{6}x_k + \frac{a}{6x_k^2}$$

仅线性收敛。

3.11 证明迭代公式

$$x_{k+1} = \frac{2}{3}x_k + \frac{a}{3x_k^2}$$

是求解方程  $(x^3 - a)^2 = 0$  的二阶方法。

3.12 试设计求  $\sqrt{a}$  的迭代公式

$$x_{k+1} = \lambda_0 x_k + \lambda_1 \left(\frac{a}{x_k}\right) + \lambda_2 \left(\frac{a^2}{x_k^3}\right)$$

使其收敛的阶尽可能高。

3.13 试设计求  $\sqrt{a}$  的迭代公式

$$x_{k+1} = \lambda_0 x_k + \lambda_1 \left(\frac{a}{x_k}\right) + \lambda_2 \left(\frac{a^3}{x_k^5}\right)$$

3.14 试设计求  $\sqrt[3]{a}$  的迭代公式

$$x_{k+1} = \lambda_0 x_k + \lambda_1 \left(\frac{a}{x_k^2}\right) + \lambda_2 \left(\frac{a^2}{x_k^5}\right)$$

3.15 早在 1225 年，有人曾求解方程  $x^3 + 2x^2 + 10x - 20 = 0$ ，并给出了高精度的实根  $x^* = 1.368\,808\,107$ ，试用牛顿法求得这个结果。

3.16 分别用牛顿法和弦截法求方程  $f(x) = x^3 + 2x^2 + 10x - 20 = 0$  的根，要求精确到  $10^{-6}$ 。

3.17 应用牛顿法于方程  $a - 1/x = 0$ ， $a > 0$ ，导出不用除法求  $1/a$  近似值的方法。

3.18 用二分法求  $x^3 + x - 4 = 0$  在  $[1, 3]$  内的近似根，要求精确到  $10^{-3}$ ，至少应二分几次？

3.19 设  $\varphi(x) = x + a(x^2 - 5)$ ，要使迭代法  $x_{k+1} = \varphi(x_k)$  局部收敛到  $x^* = \sqrt{5}$ ，求其取值范围。

3.20 试设计求解方程  $x^2 - 2x + 1 = 0$  根的牛顿迭代公式，其收敛阶为多少？



## 第4章 解线性方程组的直接法



### 学习要点

直接法就是用有限步的算术运算直接求线性方程组解的方法,主要内容有:

- (1) 消去法,包括高斯消去法和高斯-约当消去法。
- (2) 矩阵分解法,包括解一般方程组的直接三角分解法和解三对角方程组的追赶法。
- (3) 误差分析,包括条件数的概念和性质及病态方程组的判断和求解。



### 教学建议

消去法的思想是利用初等行变换,把原方程组的系数矩阵化为上三角矩阵(高斯法)或单位矩阵(高斯-约当法),这样处理既保证了方程组的解不变,又降低了算术运算次数,同时为保证算法的稳定性,每次消元必须选主元。这是本章的重点,要求学生必须掌握。矩阵分解法是消去法的另一种表现形式,特殊方程组解法的思想同矩阵分解法,但处理更为简单,算术运算次数更低。本章的难点是病态方程组的判断和误差控制,可作为选学内容。建议学时为6~10学时。

### 4.1 引言

在自然科学和工程技术中,许多问题的解决往往归结为线性方程组的求解问题,如应力分析、电学网络、分子结构、自由振动问题等。另外,许多有效的数值计算方法,其关键步骤也要解线性方程组,如三次样条插值、最小二乘法的曲线拟合,微分方程边值问题的差分法与有限元方法等。

设 $n$ 阶线性方程组

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases} \quad (4.1.1)$$

其矩阵形式为

$$Ax = b \quad (4.1.2)$$

式中,

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

由克莱姆(Gramer)法则可知,若式(4.1.2)的系数行列式 $\det(A) \neq 0$ ,则线性方程组(4.1.1)

有唯一的解。但用此方法解方程组 (4.1.1) 需计算  $n+1$  个  $n$  阶行列式, 每个  $n$  阶行列式为  $n!$  项之和, 每项又是  $n$  个元素的乘积, 需进行  $n-1$  次乘法, 所以在计算求解中仅乘法次数就高达  $(n+1)n!(n-1)$  次。当  $n$  较大时, 其计算量是相当惊人的, 这从第 1 章的引例 2 就可以知道。所以有必要研究高效率、高精度, 便于在计算机中实现的数值算法。

在实际问题中产生的线性方程组的类型有很多, 如按系数矩阵含零元素的多少进行分类, 有稠密和稀疏 (零元素占 80% 以上) 线性方程组之分; 按阶数的高低进行分类, 有高阶 (阶数在 1000 阶以上)、中阶 (500~1000 阶) 和低阶 (500 阶以下) 线性方程组之分; 按系数矩阵的形状和性质进行分类, 有对称正定、三对角、对角占优线性方程组之分等。因为数值解法必须考虑方法的计算时间和空间效率及算法的数值稳定性, 所以, 不同类型的线性方程组, 其数值解法也不相同。但是, 基本的方法可以归结为两大类: 直接法和迭代法。

所谓直接法, 就是经过有限步的算术运算, 直接求出方程组精确解的方法 (前提是计算过程中没有舍入误差)。但在实际计算中, 由于舍入误差的存在和影响, 这种方法也只能求出线性方程组的近似解。如何避免舍入误差的扩大和传播是设计直接法时必须考虑的首要问题。本章将介绍这类方法中最基本的高斯消去法和矩阵分解法。由于其具有准确性和可靠性, 因此这类方法是求解稠密线性方程组的有效方法。另外, 最近直接法在求解较高阶稀疏线性方程组方面也取得了较大的进展。

所谓迭代法, 就是用某种极限过程去逐步逼近线性方程组精确解的方法。迭代法具有占用内存少, 程序设计简单, 原始系数矩阵在计算过程中始终不变等优点。但是存在收敛性和收敛速度的问题。迭代法是求解高阶稀疏矩阵方程组的重要方法 (见第 5 章)。

## 4.2 高斯消去法

### 4.2.1 顺序高斯消去法

高斯 (Gauss) 消去法是一个古老的直接法 (早在公元前 250 年我国就掌握了解三元一次联立方程组的方法), 由它改进、变形得到的主元素消去法、矩阵三角分解法, 是目前计算机中常用于求解低阶稠密方程组的有效方法。其特点是, 通过消元将一般线性方程组的求解问题转化为上三角形方程组的求解问题。下面说明其基本思想。

#### 例 4.2.1 解线性方程组

$$\begin{cases} x_1 + x_2 + x_3 = 6 & (1) \\ x_1 + 3x_2 - 2x_3 = 1 & (2) \\ 2x_1 - 2x_2 + x_3 = 1 & (3) \end{cases}$$

**解** 首先用方程 (1) 从方程 (2) 和 (3) 中消去  $x_1$ , 即  $(1) \times (-1) + (2)$ ,  $(1) \times (-2) + (3)$ , 并保留方程 (1) 得

$$\begin{cases} x_1 + x_2 + x_3 = 6 & (1) \\ 2x_2 - 3x_3 = -5 & (4) \\ -4x_2 - x_3 = -11 & (5) \end{cases}$$

再用方程 (4) 从方程 (5) 中消去  $x_2$ , 即  $(4) \times 2 + (5)$ , 并保留方程 (1)、(4), 从而得与原方程组同解的上三角形方程组

$$\begin{cases} x_1 + x_2 + x_3 = 6 & (1) \\ 2x_2 - 3x_3 = -5 & (4) \\ -7x_3 = -21 & (6) \end{cases}$$

由方程 (6) 得  $x_3 = 3$ ，代入方程 (4) 得  $x_2 = 2$ ，再代入方程 (1) 得  $x_1 = 1$ 。

由例 4.2.1 可知，高斯消去法的基本思想是：先逐次消去变量，将原方程组变换成同解的上三角形方程组，此过程称为消元过程。然后按方程组的相反顺序求解上三角形方程组便得到原方程组的解，此过程称为回代过程。

消元过程可以用方程组增广矩阵的初等行变换来表示，例 4.2.1 中的消元过程可表示为

$$(A|b) = \begin{pmatrix} 1 & 1 & 1 & 6 \\ 1 & 3 & -2 & 1 \\ 2 & -2 & 1 & 1 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 1 & 1 & 6 \\ 0 & 2 & -3 & -5 \\ 0 & -4 & -1 & -11 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 1 & 1 & 6 \\ 0 & 2 & -3 & -5 \\ 0 & 0 & -7 & -21 \end{pmatrix}$$

一般地，设  $n$  阶线性方程组为

$$\begin{cases} a_{11}^{(0)}x_1 + a_{12}^{(0)}x_2 + \cdots + a_{1n}^{(0)}x_n = b_1^{(0)} \\ a_{21}^{(0)}x_1 + a_{22}^{(0)}x_2 + \cdots + a_{2n}^{(0)}x_n = b_2^{(0)} \\ \vdots \\ a_{n1}^{(0)}x_1 + a_{n2}^{(0)}x_2 + \cdots + a_{nn}^{(0)}x_n = b_n^{(0)} \end{cases} \quad (4.2.1)$$

其矩阵形式为

$$A^{(0)}x = b^{(0)}$$

记方程组 (4.2.1) 的增广矩阵为

$$(A^{(0)} | b^{(0)}) = \begin{pmatrix} a_{11}^{(0)} & \cdots & a_{1n}^{(0)} & b_1^{(0)} \\ \vdots & \ddots & \vdots & \vdots \\ a_{n1}^{(0)} & \cdots & a_{nn}^{(0)} & b_n^{(0)} \end{pmatrix}$$

其消元过程如下。

第 1 步：设  $a_{11}^{(0)} \neq 0$ ，记  $l_{i1} = a_{i1}^{(0)} / a_{11}^{(0)} (i = 2, 3, \dots, n)$ ，然后将  $(A^{(0)} | b^{(0)})$  的第一行的  $-l_{i1}$  倍加到第  $i$  行 ( $i = 2, 3, \dots, n$ )，得

$$(A^{(1)} | b^{(1)}) = \begin{pmatrix} a_{11}^{(0)} & a_{12}^{(0)} & \cdots & a_{1n}^{(0)} & b_1^{(0)} \\ 0 & a_{22}^{(1)} & & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{pmatrix}$$

式中，

$$\begin{aligned} a_{ij}^{(1)} &= a_{ij}^{(0)} - l_{i1}a_{1j}^{(0)} \quad (i = 2, 3, \dots, n; j = 1, 2, \dots, n) \\ b_i^{(1)} &= b_i^{(0)} - l_{i1}b_1^{(0)} \quad (i = 2, 3, \dots, n) \end{aligned}$$

即从第 2 行，第 1 列开始消元。不过按照该方法的设计，第 1 列的元素，从第 2 行到第  $n$  行必然为 0。为了减少算术运算次数，可以省略这些计算，这并不影响上三角形矩阵的元素。因此可将其改为

$$\begin{aligned} a_{ij}^{(1)} &= a_{ij}^{(0)} - l_{i1}a_{1j}^{(0)} \quad (i, j = 2, 3, \dots, n) \\ b_i^{(1)} &= b_i^{(0)} - l_{i1}b_1^{(0)} \quad (i = 2, 3, \dots, n) \end{aligned}$$

后面的步骤同理。

于是, 得到与原方程组  $\mathbf{A}^{(0)}\mathbf{x}=\mathbf{b}^{(0)}$  同解的方程组  $\mathbf{A}^{(1)}\mathbf{x}=\mathbf{b}^{(1)}$ 。

重复上述过程, 一般地, 在完成第  $k-1$  步消元后得到与原方程组  $\mathbf{A}^{(0)}\mathbf{x}=\mathbf{b}^{(0)}$  同解的方程组  $\mathbf{A}^{(k-1)}\mathbf{x}=\mathbf{b}^{(k-1)}$ , 其增广矩阵为

$$(\mathbf{A}^{(k-1)}|\mathbf{b}^{(k-1)})=\begin{pmatrix} a_{11}^{(0)} & \cdots & a_{1k}^{(0)} & \cdots & a_{1n}^{(0)} & b_1^{(0)} \\ & \ddots & \vdots & \ddots & \vdots & \vdots \\ & & a_{kk}^{(k-1)} & \cdots & a_{kn}^{(k-1)} & b_k^{(k-1)} \\ & & \vdots & \ddots & \vdots & \vdots \\ & & a_{nk}^{(k-1)} & \cdots & a_{nn}^{(k-1)} & b_n^{(k-1)} \end{pmatrix}$$

第  $k$  步: 设  $a_{kk}^{(k-1)} \neq 0$  时, 记  $l_{ik}=a_{ik}^{(k-1)}/a_{kk}^{(k-1)} (i=k+1, k+2, \cdots, n)$ , 然后将  $(\mathbf{A}^{(k-1)}|\mathbf{b}^{(k-1)})$  的第  $k$  行的  $-l_{ik}$  倍加到第  $i$  行  $(i=k+1, k+2, \cdots, n)$ , 得

$$(\mathbf{A}^{(k)}|\mathbf{b}^{(k)})=\begin{pmatrix} a_{11}^{(0)} & \cdots & a_{1k}^{(0)} & a_{1(k+1)}^{(0)} & \cdots & a_{1n}^{(0)} & b_1^{(0)} \\ & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ & & a_{kk}^{(k-1)} & a_{k(k+1)}^{(k-1)} & \cdots & a_{kn}^{(k-1)} & b_k^{(k-1)} \\ & & & a_{(k+1)(k+1)}^{(k)} & \cdots & a_{(k+1)n}^{(k)} & b_{k+1}^{(k)} \\ & & & \vdots & \ddots & \vdots & \vdots \\ & & & a_{n(k+1)}^{(k)} & \cdots & a_{nn}^{(k)} & b_n^{(k)} \end{pmatrix}$$

式中

$$\begin{aligned} a_{ij}^{(k)} &= a_{ij}^{(k-1)} - l_{ik}a_{kj}^{(k-1)} & (i, j = k+1, k+2, \cdots, n) \\ b_i^{(k)} &= b_i^{(k-1)} - l_{ik}b_k^{(k-1)} & (i = k+1, k+2, \cdots, n) \end{aligned}$$

如此继续, 经  $n-1$  步消元后, 可得原方程组的同解三角形方程组  $\mathbf{A}^{(n-1)}\mathbf{x}=\mathbf{b}^{(n-1)}$ , 即

$$\begin{pmatrix} a_{11}^{(0)} & a_{12}^{(0)} & \cdots & a_{1n}^{(0)} \\ & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ & & \ddots & \vdots \\ & & & a_{nn}^{(n-1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(0)} \\ b_2^{(1)} \\ \vdots \\ b_n^{(n-1)} \end{pmatrix}$$

综上所述, 整个消元过程如下。

对  $k=1, 2, \cdots, n-1$  逐次计算

$$\begin{cases} l_{ik} = a_{ik}^{(k-1)} / a_{kk}^{(k-1)} (i = k+1, \cdots, n) \\ a_{ij}^{(k)} = a_{ij}^{(k-1)} - l_{ik}a_{kj}^{(k-1)} (i, j = k+1, \cdots, n) \\ b_i^{(k)} = b_i^{(k-1)} - l_{ik}b_k^{(k-1)} (i = k+1, \cdots, n) \end{cases} \quad (4.2.2)$$

回代过程如下。

逐步回代求得原方程组的解

$$\begin{cases} x_n = b_n^{(n-1)} / a_{nn}^{(n-1)} \\ x_k = (b_k^{(k-1)} - \sum_{j=k+1}^n a_{kj}^{(k-1)}x_j) / a_{kk}^{(k-1)} (k = n-1, n-2, \cdots, 1) \end{cases} \quad (4.2.3)$$

由以上计算过程可知, 消元过程能进行到底的条件是主元素  $a_{kk}^{(k-1)} \neq 0 (k=1, 2, \cdots, n-1)$ , 那么, 当系数矩阵  $\mathbf{A}$  具有什么特征时, 才能保证主元素不为零呢? 下面的定理给出了主元素不为零的一个条件。

**定理 4.2.1** 如果  $n$  阶矩阵  $\mathbf{A}$  的顺序主子式均不为零, 即

$$a_{11} \neq 0, \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \neq 0, \dots, \det(\mathbf{A}) \neq 0$$

则用高斯消去法求解方程组  $\mathbf{Ax} = \mathbf{b}$  时的主元素为

$$a_{kk}^{(k-1)} \neq 0 \quad (k=1, 2, \dots, n)$$

(证明略。)

下面分析高斯消去法的计算量：由于在计算机中进行乘除运算所需的时间远大于进行加减运算所需的时间，故只考虑进行乘除法的计算量。

由消元过程式 (4.2.2) 可知，第  $k$  次消元求  $l_{ik}$  需进行  $n-k$  次除法，求  $a_{ij}^{(k)}$  需进行  $(n-k)^2$  次乘法，求  $b_i^{(k)}$  需进行  $n-k$  次乘法，所以，消元过程中乘、除法总计算量为

$$\begin{aligned} \sum_{k=1}^{n-1} [(n-k) + (n-k)^2 + (n-k)] &= 2 \sum_{k=1}^{n-1} (n-k) + \sum_{k=1}^{n-1} (n-k)^2 \\ &= 2[(n-1) + (n-2) + \dots + 2 + 1] + [(n-1)^2 + (n-2)^2 + \dots + 2^2 + 1^2] \\ &= 2 \times \frac{1}{2} n(n-1) + \frac{1}{6} n(n-1)(2n-1) \\ &= \frac{1}{3} n^3 + \frac{1}{2} n^2 - \frac{5}{6} n \end{aligned}$$

由回代过程式 (4.2.3) 可知，求  $x_k$  需进行 1 次除法和  $(n-k)$  次乘法，所以回代过程的乘除法的计算量为

$$\sum_{k=1}^n (n-k) + n = \frac{1}{2} n(n-1) + n = \frac{1}{2} n^2 + \frac{1}{2} n$$

因此，高斯消去法的乘、除法总计算量为

$$\frac{1}{3} n^3 + \frac{1}{2} n^2 - \frac{6}{5} n + \frac{1}{2} n^2 + \frac{1}{2} n = \frac{1}{3} n^3 + n^2 - \frac{1}{3} n$$

可以通过表 4.2.1 比较高斯消去法与克莱姆法则的乘、除法总计算量  $(n+1)n!(n-1) + n$ 。

表 4.2.1 总计算量比较

方程组的阶数	3	10	20	50
高斯消去法	17	430	3060	44 150
克莱姆法则	51	359 251 210	$9.7 \times 10^{20}$	$7.9 \times 10^{67}$

使用每秒能完成 12.5 万次乘、除法的计算机，用高斯消去法求 20 阶线性方程组所需的时间为

$$\frac{3060}{1.25 \times 10^5} \approx 0.02448\text{s}$$

大约为 0.02 秒，而由例 1.1.2 可知，用克莱姆法则求同一问题则需 2 亿 4 千万年，这进一步说明了计算方法的重要性。

**例 4.2.2** 用高斯消去法求解线性方程组

$$\begin{cases} 2x_1 + 2x_2 + 3x_3 = 3 \\ 4x_1 + 7x_2 + 7x_3 = 1 \\ -2x_1 + 4x_2 + 5x_3 = -7 \end{cases}$$

**解** 只需对增广矩阵进行初等变换后化为上三角形方程组，回代求解即可。

$$(\mathbf{A}|\mathbf{b}) = \begin{pmatrix} 2 & 2 & 3 & 3 \\ 4 & 7 & 7 & 1 \\ -2 & 4 & 5 & -7 \end{pmatrix} \rightarrow \begin{pmatrix} 2 & 2 & 3 & 3 \\ 0 & 3 & 1 & -5 \\ 0 & 6 & 8 & -4 \end{pmatrix} \rightarrow \begin{pmatrix} 2 & 2 & 3 & 3 \\ 0 & 3 & 1 & -5 \\ 0 & 0 & 6 & 6 \end{pmatrix}$$

故 
$$\begin{cases} 2x_1 + 2x_2 + 3x_3 = 3 \\ 3x_2 + x_3 = -5 \\ 6x_3 = 6 \end{cases}$$

回代求解得 
$$x_3 = 1, \quad x_2 = \frac{-5 - x_3}{3} = -2, \quad x_1 = \frac{3 - 3x_3 - 2x_2}{2} = 2$$

**例 4.2.3** 用高斯消去法求方程组

$$\begin{cases} 2x_1 + x_2 + x_3 = 4 \\ x_1 + 3x_2 + 2x_3 = 6 \\ x_1 + 2x_2 + 2x_3 = 5 \end{cases}$$

的解，并求系数行列式  $\det(\mathbf{A})$ 。

**解** 对增广矩阵实施高斯消元过程，有

$$(\mathbf{A}|\mathbf{b}) = \begin{pmatrix} 2 & 1 & 1 & 4 \\ 1 & 3 & 2 & 6 \\ 1 & 2 & 2 & 5 \end{pmatrix} \rightarrow \begin{pmatrix} 2 & 1 & 1 & 4 \\ 0 & 5/2 & 3/2 & 4 \\ 0 & 3/2 & 3/2 & 3 \end{pmatrix} \rightarrow \begin{pmatrix} 2 & 1 & 1 & 4 \\ & 5/2 & 3/2 & 4 \\ & & 3/5 & 3/5 \end{pmatrix}$$

回代求解得

$$x_1 = x_2 = x_3 = 1, \quad \det(\mathbf{A}) = 2 \times \frac{5}{2} \times \frac{3}{5} = 3$$

## 4.2.2 主元素高斯消去法

主元素高斯消去法是为控制舍入误差的扩大和传播而提出的。在顺序高斯消去法的消元过程中，若出现  $a_{kk}^{(k-1)} = 0$ ，则消元无法进行，即使  $a_{kk}^{(k-1)} \neq 0$ 。但是如果其绝对值很小的话，把它作为除数，就会导致其他元素量级的巨大增长和舍入误差的扩大，最后使计算结果失真。

**例 4.2.4** 解线性方程组

$$\begin{cases} 0.0001x_1 + x_2 = 1 \\ x_1 + x_2 = 2 \end{cases}$$

**解** 容易验证，方程组的准确解为

$$x_1 = 10000/9999, \quad x_2 = 9998/9999$$

用 3 位十进制浮点数求解，把第二个方程中的  $x_1$  消去，得到

$$\begin{cases} 0.0001x_1 + x_2 = 1 \\ -10000x_2 = -10000 \end{cases}$$

由第二个方程解出  $x_2 = 1$ ，代入第一个方程得  $x_1 = 0$ 。近似解与精确解相差很大，这是因为用 0.0001 作除数，使舍入误差剧增所造成的。为了控制舍入误差，先将方程组中的两个方程交换一下，变为

$$\begin{cases} x_1 + x_2 = 2 \\ 0.0001x_1 + x_2 = 1 \end{cases}$$

消去第二个方程中的  $x_1$ ，得

$$\begin{cases} x_1 + x_2 = 2 \\ x_2 = 1 \end{cases}$$

从而求得  $x_1 = x_2 = 1$ ，结果与准确解非常接近。

事实上，从顺序高斯消去法的消元过程式 (4.2.2) 可以看出，当  $|l_{ik}| = \left| \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} \right| > 1$  时，若  $a_{kj}^{(k-1)}$  有舍入误差  $e_{k-1}$ ，则经过计算  $a_{ij}^{(k)} = a_{ij}^{(k-1)} - l_{ik} a_{kj}^{(k-1)}$ ， $a_{ij}^{(k)}$  就会有比  $e_{k-1}$  更大的舍入误差  $e_k = |l_{ik}| e_{k-1}$  产生，同样  $b_i^{(k)}$  的误差也会比  $b_k^{(k-1)}$  的舍入误差大。因此，为了达到减小舍入误差的目的，必须保证  $|l_{ik}| < 1$ ，即  $|a_{kk}^{(k-1)}| > |a_{ik}^{(k-1)}| (i = k+1, k+2, \dots, n)$ ，于是可以对顺序消去法进行改进。用交换方程或交换未知数次序的方法，选择绝对值尽可能大的系数作为第  $k$  步消元过程中的主元素  $a_{kk}^{(k-1)}$ ，这就是主元素消去法的基本思想。由于选取主元的范围不同，相应地，就有不同的主元消去法。在第  $k$  步消去过程中，如果从  $a_{kk}^{(k-1)}$  的右下方子矩阵中选取主元素，则称为全主元消去法；如果从  $a_{kk}^{(k-1)}$  所在列的下方选取主元素，则称为列主元消去法。由于列主元消去法选主元的范围小，不改变未知数的次序，且能保证  $|l_{ik}| < 1$ ，使消元法稳定，因而被广泛使用。

在计算机中求解方程组  $Ax = b$  时，常把右端常数项  $b$  作为系数矩阵  $A$  的第  $n+1$  列，增广矩阵仍然记为  $A$ 。列主元消去法的算法框图如图 4.2.1 所示。

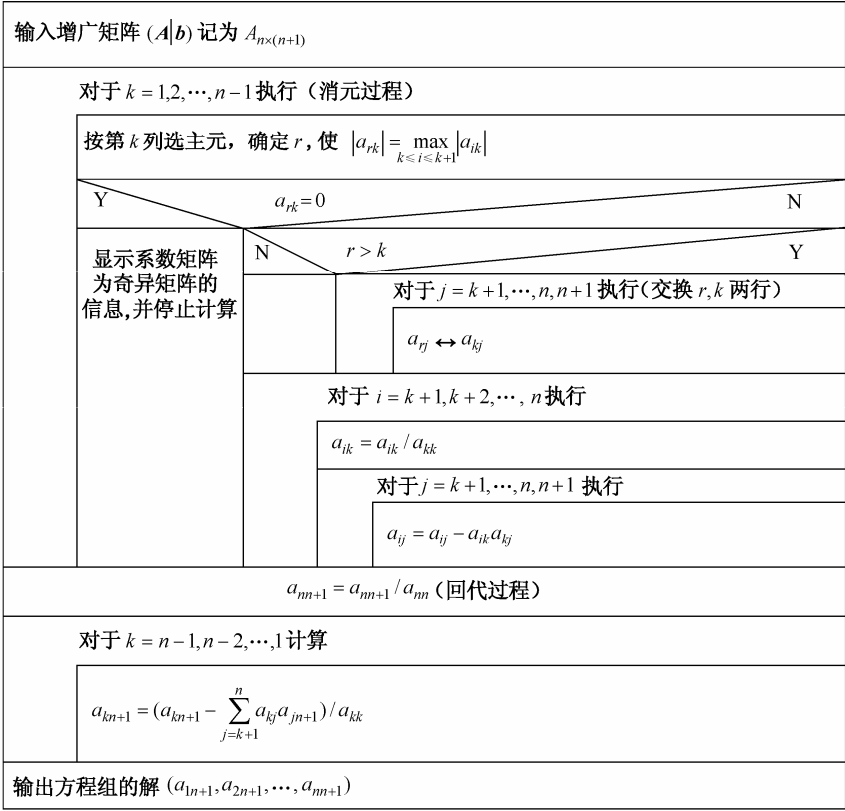


图 4.2.1 列主元高斯消去法的算法框图

#### 例 4.2.5 用列主元素消去法解方程组

$$\begin{cases} 12x_1 - 3x_2 + 3x_3 = 15 \\ -18x_1 + 3x_2 - x_3 = -15 \\ x_1 + x_2 + x_3 = 6 \end{cases}$$

解

$$(A|b) = \begin{pmatrix} 12 & -3 & 3 & 15 \\ -18 & 3 & -1 & -15 \\ 1 & 1 & 1 & 6 \end{pmatrix} \xrightarrow{1,2\text{行交换}} \begin{pmatrix} -18 & 3 & -1 & -15 \\ 12 & -3 & 3 & 15 \\ 1 & 1 & 1 & 6 \end{pmatrix} \xrightarrow{\text{消元}} \begin{pmatrix} -18 & 3 & -1 & -15 \\ 0 & -1 & 7/3 & 5 \\ 0 & 7/6 & 17/18 & 31/6 \end{pmatrix}$$

$$\xrightarrow{2,3\text{行交换}} \begin{pmatrix} -18 & 3 & -1 & -15 \\ 0 & 7/6 & 17/18 & 31/6 \\ 0 & -1 & 7/3 & 5 \end{pmatrix} \xrightarrow{\text{消元}} \begin{pmatrix} -18 & 3 & -1 & -15 \\ 0 & 7/6 & 17/18 & 31/6 \\ 0 & 0 & 22/7 & 66/7 \end{pmatrix}$$

回代得解

$$x_3 = 3, x_2 = 2, x_1 = 1$$

### 4.2.3 高斯-约当消去法

高斯消去法的基本思想是消去对角线下方的元素，使系数矩阵变为上三角矩阵。现将这一方法进行修正，即消去对角线下方和上方的元素，使系数矩阵变为单位矩阵，这种方法称为高斯-约当（Gauss-Jordan）消去法。它不需要回代过程。

#### 例 4.2.6 解线性方程组

$$\begin{cases} x_1 + x_2 - x_3 = 1 \\ x_1 + 2x_2 - 2x_3 = 0 \\ -2x_1 + x_2 + x_3 = 1 \end{cases}$$

解 对方程组的增广矩阵实施消元过程，将系数矩阵对角化为

$$(A|b) = \begin{pmatrix} 1 & 1 & -1 & 1 \\ 1 & 2 & -2 & 0 \\ -2 & 1 & 1 & 1 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 1 & -1 & 1 \\ 0 & 1 & -1 & -1 \\ 0 & 3 & -1 & 3 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & -1 & -1 \\ 0 & 0 & 2 & 6 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{pmatrix}$$

故

$$\begin{cases} x_1 = 2 \\ x_2 = 2 \\ x_3 = 3 \end{cases}$$

与讨论高斯消去法相类似，高斯-约当消去法的计算过程如下。

对于  $k=1, 2, \dots, n$ ，计算

$$\begin{cases} a_{kj}^{(k)} = a_{kj}^{(k-1)} / a_{kk}^{(k-1)} (j = k+1, k+2, \dots, n+1) \\ a_{ij}^{(k)} = a_{ij}^{(k-1)} - a_{ik}^{(k-1)} \times a_{kj}^{(k)} (i = 1, 2, \dots, n \text{ 且 } i \neq k; j = k+1, k+2, \dots, n+1) \end{cases}$$

在每步消元过程中，选列主元，可得到列主元高斯-约当消去法，其算法框图如图 4.2.2 所示。

该方法的计算量统计如下：

乘法次数为

$$\begin{aligned} & n(n-1) + (n-1)(n-1) + \dots + (n-1) \times 1 \\ &= (n-1) \sum_{k=1}^n k = (n-1) \frac{1}{2} n(n+1) = \frac{1}{2} n^3 - \frac{1}{2} n \end{aligned}$$



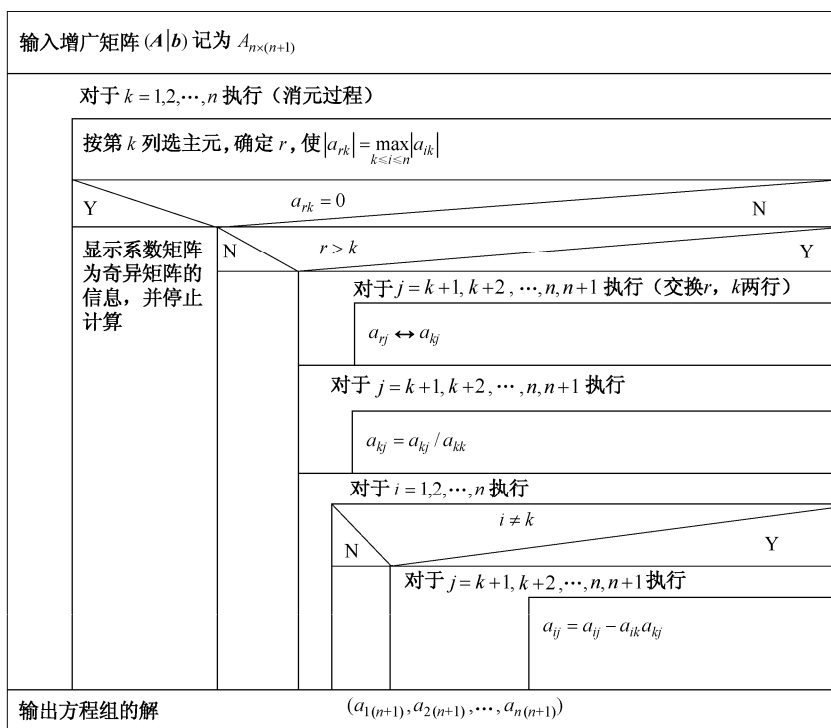


图 4.2.2 列主元高斯-约当消去法算法框图

除法次数为

$$n + (n-1) + \dots + 1 = \frac{1}{2}n^2 + \frac{1}{2}n$$

乘除法的总次数为

$$\frac{1}{2}n^3 + \frac{1}{2}n^2$$

它比高斯消去法的计算量大, 但不需要回代过程, 算法的结构略为简单, 并且比较适合求一个矩阵的逆矩阵。

例 4.2.7 用主元素高斯-约当消去法解方程组。

$$\begin{cases} x_2 + x_3 = 1 \\ x_1 + 2x_2 + 3x_3 = 0 \\ x_1 + 4x_2 + 6x_3 = 2 \end{cases}$$

解

$$\begin{aligned} (A|b) &= \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 2 & 3 & 0 \\ 1 & 4 & 6 & 2 \end{pmatrix} \xrightarrow{1,2\text{行交换}} \begin{pmatrix} 1 & 2 & 3 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 4 & 6 & 2 \end{pmatrix} \xrightarrow{\text{消元}} \begin{pmatrix} 1 & 2 & 3 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 2 & 3 & 2 \end{pmatrix} \\ &\xrightarrow{2,3\text{行交换}} \begin{pmatrix} 1 & 2 & 3 & 0 \\ 0 & 2 & 3 & 2 \\ 0 & 1 & 1 & 1 \end{pmatrix} \xrightarrow{\text{消元}} \begin{pmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 3/2 & 1 \\ 0 & 0 & -1/2 & 0 \end{pmatrix} \xrightarrow{\text{消元}} \begin{pmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{aligned}$$

所以方程组的解为

$$x_1 = -2, \quad x_2 = 1, \quad x_3 = 0$$

例 4.2.8 用列主元高斯-约当消去法求  $A$  的逆矩阵。

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{pmatrix}$$

解 将矩阵  $A$  和单位矩阵  $I$  写在一起构成  $n \times 2n$  阶矩阵  $(A, I) = C$ ，然后对其进行选主元和消元，得

$$\begin{aligned} C = \begin{pmatrix} 1 & 2 & 3 & 1 & 0 & 0 \\ 2 & 4 & 5 & 0 & 1 & 0 \\ 3 & 5 & 6 & 0 & 0 & 1 \end{pmatrix} &\xrightarrow{\text{选主元}} \begin{pmatrix} 3 & 5 & 6 & 0 & 0 & 1 \\ 2 & 4 & 5 & 0 & 1 & 0 \\ 1 & 2 & 3 & 1 & 0 & 0 \end{pmatrix} \xrightarrow{\text{消元}} \begin{pmatrix} 1 & 5/3 & 2 & 0 & 0 & 1/3 \\ 0 & 2/3 & 1 & 0 & 1 & -2/3 \\ 0 & 1/3 & 1 & 1 & 0 & -1/3 \end{pmatrix} \\ &\xrightarrow{\text{消元}} \begin{pmatrix} 1 & 0 & -1/2 & 0 & -5/2 & 2 \\ 0 & 1 & 3/2 & 0 & 3/2 & -1 \\ 0 & 0 & 1/2 & 1 & -1/2 & 0 \end{pmatrix} \xrightarrow{\text{消元}} \begin{pmatrix} 1 & 0 & 0 & 1 & -3 & 2 \\ 0 & 1 & 0 & -3 & 3 & -1 \\ 0 & 0 & 1 & 2 & -1 & 0 \end{pmatrix} = (I, A^{-1}) \end{aligned}$$

所以

$$A^{-1} = \begin{pmatrix} 1 & -3 & 2 \\ -3 & 3 & -1 \\ 2 & -1 & 0 \end{pmatrix}$$

## 4.3 矩阵三角分解法

### 4.3.1 高斯消去法与矩阵三角分解

如果用矩阵形式表示，高斯消去法的消元过程是对方程组 (4.2.1) 的增广矩阵进行一系列的初等行变换后，将系数矩阵  $A$  化成上三角矩阵的过程，等价于用一系列的初等变换矩阵左乘增广矩阵。因此，消元过程可以通过矩阵运算来实现。

设  $a_{11} \neq 0, l_{i1} = a_{i1}^{(0)} / a_{11}^{(0)} (i = 2, 3, \dots, n)$ ，记

$$L_1 = \begin{pmatrix} 1 & & & \\ -l_{21} & 1 & & \\ \vdots & & \ddots & \\ -l_{n1} & & & 1 \end{pmatrix}$$

高斯消去法的第一步消元相当于用初等矩阵  $L_1$  左乘增广矩阵  $(A^{(0)}, b^{(0)})$ ，记为

$$L_1(A^{(0)}, b^{(0)}) = (A^{(1)}, b^{(1)})$$

设  $a_{kk}^{(k-1)} \neq 0, l_{ik} = a_{ik}^{(k-1)} / a_{kk}^{(k-1)} (i = k+1, k+2, \dots, n)$ ，记

$$L_k = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & 1 & \\ & & & -l_{k+1,k} & 1 & \\ & & & \vdots & & \ddots \\ & & & -l_{n,k} & & & 1 \end{pmatrix}$$

第  $k$  次消元相当于用初等矩阵  $L_k$  左乘增广矩阵  $(A^{(k-1)}, b^{(k-1)})$ , 记为

$$L_k(A^{(k-1)}, b^{(k-1)}) = (A^{(k)}, b^{(k)})$$

重复这一过程, 经过  $n-1$  步消元后, 最后得到

$$L_{n-1}L_{n-2}\cdots L_1(A^{(0)}, b^{(0)}) = (A^{(n-1)}, b^{(n-1)})$$

因为  $L_k$  ( $k=1, 2, \dots, n-1$ ) 均为非奇异矩阵, 故它们的逆矩阵存在。

容易求出

$$L_k^{-1} = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & l_{k+1,k} & 1 & & \\ & \vdots & & \ddots & \\ & l_{n,k} & & & 1 \end{pmatrix}$$

于是

$$(A^{(0)}, b^{(0)}) = L_1^{-1}L_2^{-1}\cdots L_{n-1}^{-1}(A^{(n-1)}, b^{(n-1)})$$

可求得

$$L = L_1^{-1}L_2^{-1}\cdots L_{n-1}^{-1} = \begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{n-1} & 1 \end{pmatrix}$$

记

$$U = A^{(n-1)}, Y = b^{(n-1)}$$

于是, 有

$$(A^{(0)}, b^{(0)}) = L(A^{(n-1)}, b^{(n-1)}) = (LU, LY)$$

这说明, 在  $a_{kk}^{(k-1)} \neq 0$  ( $k=1, 2, \dots, n-1$ ) 的条件下, 高斯消去法的消元过程实质上是把系数矩阵  $A$  分解成一个单位下三角矩阵  $L$  与上三角矩阵  $U$  的乘积的过程, 于是由定理 4.2.1 可得如下定理。

**定理 4.3.1** 设  $A$  为  $n$  阶矩阵, 如果  $A$  的顺序主子式  $D_i \neq 0$  ( $i=1, 2, \dots, n$ ), 则  $A$  可分解为一个单位下三角矩阵  $L$  和一个上三角矩阵  $U$  的乘积, 且这种分解是唯一的。

当系数矩阵完成三角分解后, 对于求解方程组  $Ax = b$ , 就有

$$Ax = b \xrightarrow{A=LU} LUx = b \xrightarrow{\text{令 } y=Ux} \begin{cases} Ly = b \\ Ux = y \end{cases}$$

高斯消去法的消元过程相当于分解  $A = LU$  及求解三角形方程组  $Ly = b$ , 回代过程则是求解另一个三角形方程组  $Ux = y$ , 因此, 解方程组的问题可转化为矩阵的三角分解问题。

### 4.3.2 直接三角分解法

由定理 4.3.1 知, 当方阵  $A$  的各阶顺序主子式都不为零时,  $A$  可唯一地分解为  $A = LU$ , 式中  $L$  为单位下三角矩阵,  $U$  为上三角矩阵。本节讨论直接从系数矩阵  $A$  的元素得到计算矩阵  $L, U$  的元素的递推公式, 而不需要高斯消去法的任何中间步骤, 因而称为直接三角分解法, 也称为 Doolittle 分解。

由于两个矩阵相等就是它们的对应元素相等, 因此通过比较  $A$  与  $LU$  的对应元素, 即可得到直接计算  $L, U$  的元素的公式。

设

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ & u_{22} & u_{23} & \cdots & u_{2n} \\ & & u_{33} & \cdots & u_{3n} \\ & & & \ddots & \vdots \\ & & & & u_{nn} \end{pmatrix} = LU \quad (4.3.1)$$

利用矩阵乘法规则，并比较式(4.3.1)的两端元素，步骤如下。

第1步：由 $L$ 的第1行和 $U$ 的第 $j$ 列元素相乘后与 $A$ 的对应元素相等，有 $a_{1j} = u_{1j}$ ，从而得到 $U$ 的第一行元素

$$u_{1j} = a_{1j} \quad (j=1, 2, \cdots, n)$$

由 $L$ 的第 $i$ 行和 $U$ 的第1列元素相乘后与 $A$ 的对应元素相等，有 $a_{i1} = l_{i1}u_{11}$ ，从而得到 $L$ 的第1列元素

$$l_{i1} = a_{i1}/u_{11} \quad (i=2, 3, \cdots, n)$$

这样第1步就确定出了 $U$ 的第1行元素和 $L$ 的第1列元素。

第2步：(与第1步类似)

由 $a_{2j} = l_{21}u_{1j} + u_{2j}$ 得 $U$ 的第2行元素

$$u_{2j} = a_{2j} - l_{21}u_{1j} \quad (j=2, 3, \cdots, n)$$

由 $a_{i2} = l_{i1}u_{12} + l_{i2}u_{22}$ 得 $L$ 的第2列元素

$$l_{i2} = (a_{i2} - l_{i1}u_{12})/u_{22} \quad (i=3, 4, \cdots, n)$$

从而确定出了 $U$ 的第2行元素和 $L$ 的第2列元素

假设已经确定 $U$ 的 $k-1$ 行及 $L$ 的 $k-1$ 列( $1 \leq k \leq n$ )的全部元素。

第 $k$ 步：

由 $a_{kj} = \sum_{r=1}^n l_{kr}u_{rj}$  ( $j \leq k$ )，注意 $L$ 为单位下三角矩阵，当 $r > k$ 时， $l_{kr} = 0$ ，而 $l_{kk} = 1$ ，则

$$a_{kj} = \sum_{r=1}^{k-1} l_{kr}u_{rj} + u_{kj}$$

从而得到

$$u_{kj} = a_{kj} - \sum_{r=1}^{k-1} l_{kr}u_{rj} \quad (j=k, k+1, \cdots, n)$$

即得到 $U$ 的第 $k$ 行元素。

同理，由 $a_{ik} = \sum_{r=1}^n l_{ir}u_{rk}$  ( $i > k$ )，注意 $U$ 为上三角矩阵，当 $r > k$ 时， $u_{rk} = 0$ ，则

$$a_{ik} = \sum_{r=1}^k l_{ir}u_{rk} = \sum_{r=1}^{k-1} l_{ir}u_{rk} + l_{ik}u_{kk}$$

从而得到

$$l_{ik} = (a_{ik} - \sum_{r=1}^{k-1} l_{ir}u_{rk})/u_{kk} \quad (i=k+1, k+2, \cdots, n)$$

即得到 $L$ 的第 $k$ 列元素。

上述步骤共进行 $n$ 步，就可以确实出 $L$ 及 $U$ 的全部元素，完成了矩阵 $A$ 的LU分解，计算过程归纳如下。

对于  $k=1,2,\cdots,n$ , 计算

$$\begin{cases} u_{kj} = a_{kj} - \sum_{r=1}^{k-1} l_{kr} u_{rj} \quad (j = k, k+1, \cdots, n) \\ l_{ik} = (a_{ik} - \sum_{r=1}^{k-1} l_{ir} u_{rk}) / u_{kk} \quad (i = k+1, k+2, \cdots, n) \end{cases} \quad (4.3.2)$$

(定义  $\sum_1^0 = 0$ )

在计算和存储  $L, U$  的元素时, 可采用如图 4.3.1 所示的方式。

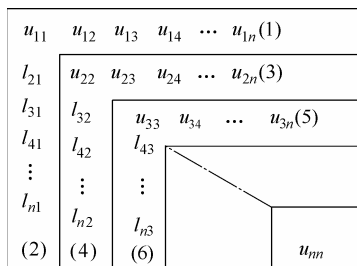


图 4.3.1  $L, U$  的计算和存储

从式 (4.3.2) 可以看出, 在算出  $u_{kj}$  后  $a_{kj}$  不再有用, 在算出  $l_{ik}$  后  $a_{ik}$  不再有用, 因此, 编程时, 可将  $L, U$  的元素放入  $A$  的相应元素位置中。

当系数矩阵  $A$  完成了  $A = LU$  分解后, 这时方程组  $Ax = b$  就化为  $L(Ux) = b$ , 它等价于求解两个方程组  $Ly = b$  与  $Ux = y$ , 具体计算公式为

$$\begin{cases} y_1 = b_1 \\ y_k = b_k - \sum_{j=1}^{k-1} l_{kj} y_j \quad (k = 2, 3, \cdots, n) \end{cases} \quad (4.3.3)$$

$$\begin{cases} x_n = y_n / u_{nn} \\ x_k = (y_k - \sum_{j=k+1}^n u_{kj} x_j) / u_{kk}, \quad (k = n-1, n-2, \cdots, 1) \end{cases} \quad (4.3.4)$$

例 4.3.1 用直接三角分解法解方程组。

$$\begin{pmatrix} 2 & 1 & 5 \\ 4 & 1 & 12 \\ -2 & -4 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 11 \\ 27 \\ 12 \end{pmatrix}$$

解 利用式 (4.3.2) 计算如下。

第 1 步: 计算  $U$  的第 1 行及  $L$  的第 1 列元素

$$\begin{aligned} u_{11} &= 2, u_{12} = 1, u_{13} = 5 \\ l_{21} &= a_{21} / u_{11} = 2, l_{31} = a_{31} / u_{11} = -1 \end{aligned}$$

第 2 步: 计算  $U$  的第 2 行及  $L$  的第 2 列元素

$$\begin{aligned} u_{22} &= a_{22} - l_{21} u_{12} = -1, u_{23} = a_{23} - l_{21} u_{13} = 2 \\ l_{32} &= (a_{32} - l_{31} u_{12}) / u_{22} = 3 \end{aligned}$$

第 3 步: 计算  $U$  的第 3 行

$$u_{33} = a_{33} - (l_{31} u_{13} + l_{32} u_{23}) = 4$$

故

$$A = LU = \begin{pmatrix} 1 & & \\ 2 & 1 & \\ -1 & 3 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & 5 \\ & -1 & 2 \\ & & 4 \end{pmatrix}$$

利用式 (4.3.3) 解  $Ly = b$  得

$$y_1 = 11, y_2 = 5, y_3 = 8$$

用式 (4.3.4) 解  $Ux = y$  得

$$x_3 = 2, x_2 = -1, x_1 = 1$$

矩阵的三角分解可按如图 4.3.2 所示的紧凑格式来表示, 这种格式便于记忆和计算。

$(a_{11})u_{11}$	$(a_{12})u_{12}$	$(a_{13})u_{13}$	$\cdots$	$(a_{1n})u_{1n}$	$(b_1)y_1$
$(a_{21})l_{21}$	$(a_{22})u_{22}$	$(a_{23})u_{23}$	$\cdots$	$(a_{2n})u_{2n}$	$(b_2)y_2$
$(a_{31})l_{31}$	$(a_{32})l_{32}$	$(a_{33})u_{33}$	$\cdots$	$(a_{3n})u_{3n}$	$(b_3)y_3$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	
$(a_{n1})l_{n1}$	$(a_{n2})l_{n2}$	$(a_{n3})l_{n3}$	$\cdots$	$u_{nn}$	$(b_n)y_n$

图 4.3.2 紧凑格式表示的矩阵三角分解

① 计算顺序: 将  $a_{ij}, u_{ij}, l_{ij}, b_i, y_i$  按图 4.3.2 列好, 计算时从外到内进行。每个框中, 先计算行, 从左到右依次计算  $u_{kj}, y_k$ , 后计算列, 从上到下计算  $l_{ik}$ 。

② 计算方法: 按行计算时, 需将所求元素  $u_{kj}(y_k)$  的对应元素  $a_{kj}(b_k)$  逐次减去  $u_{kj}(y_k)$  所在行左边框的元素  $l_{ki}$  乘以  $u_{kj}(y_k)$  所在列上面各框相应元素  $u_{ij}(y_j)$ 。按列计算  $l_{ik}$  时, 在进行上述计算后还要除以  $l_{ik}$  所在框的对角元素  $u_{kk}$ 。

例 4.3.2 用紧凑格式解现行方程组。

$$\begin{cases} 3x_1 + 2x_2 + 5x_3 = 6 \\ -x_1 + 4x_2 + 3x_3 = 5 \\ x_1 - x_2 + 3x_3 = 1 \end{cases}$$

解 按图 4.3.2 计算, 结果如图 4.3.3 所示。

(3)3	(2)2	(5)5	(6)6
$(-1) - \frac{1}{3}$	$(4)4 + \frac{2}{3} = \frac{14}{3}$	$(3)3 + \frac{5}{3} = \frac{14}{3}$	$(5)5 + 2 = 7$
$(1)\frac{1}{3}$	$(-1)(-1 - \frac{1}{3} \times 2) / \frac{14}{3} = \frac{-5}{14}$	$(3)3 + \frac{5}{3} - \frac{5}{3} = 3$	$(1)1 + \frac{5}{2} - 2 = \frac{3}{2}$

图 4.3.3 例 4.3.2 计算结果

所以

$$L = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{3} & 1 & 0 \\ \frac{1}{3} & -\frac{5}{14} & 1 \end{pmatrix}, U = \begin{pmatrix} 3 & 2 & 5 \\ 0 & \frac{14}{3} & \frac{14}{3} \\ 0 & 0 & 3 \end{pmatrix}, y = \begin{pmatrix} 6 \\ 7 \\ \frac{3}{2} \end{pmatrix}$$

解方程组  $Ux = y$ , 即

$$\begin{pmatrix} 3 & 2 & 5 \\ 0 & \frac{14}{3} & \frac{14}{3} \\ 0 & 0 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 6 \\ 7 \\ \frac{3}{2} \end{pmatrix}$$

得原方程组的解为

$$x_3 = \frac{1}{2}, \quad x_2 = 1, \quad x_1 = \frac{1}{2}$$

关于直接三角分解法的几点说明:

① 用直接三角分解法求线性方程式组  $Ax = b$  的乘、除法的计算量也是  $n^3/3$  数量级, 与高斯消去法解同阶方程组所需计算量基本相同。

② 由于在求出  $u_{ij}, l_{ij}, y_i$  后,  $a_{ij}$  和  $b_i$  无须保留, 故编程计算时, 可把  $L, U$  和  $y$  存在  $A, b$  所在的单元, 回代时,  $x$  取代  $y$ , 因此, 整个计算过程中不需要增加新的存储单元。

③ 从三角分解的计算公式 (4.3.2) 可以看出, 系数矩阵的三角分解与右端项无关, 因而在计算多个系数矩阵相同而右端项不同的一系列线性方程组时, 用直接三角分解法更为简便。

④ 完成  $A = LU$  分解后, 可以较容易地求出行列式

$$\det(A) = \det(L) \times \det(U) = u_{11}u_{22} \cdots u_{nn}$$

⑤ 直接三角分解法一般地可采用选主元的技术, 以便使算法更加稳定。

⑥ 也可以把系数矩阵  $A$  分解成一个下三角矩阵与一个单位上三角矩阵的乘积, 矩阵的这一分解称为 Crout 分解。

## 4.4 解三对角方程组的追赶法

在三次样条插值问题和常微分方程边值问题的数值计算方法中, 都要求解对角占优的三对角线性方程组

$$Ax = f \quad (4.4.1)$$

式中

$$A = \begin{pmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{pmatrix}, \quad f = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} \quad (4.4.2)$$

并且  $A$  满足条件

$$\begin{cases} |b_1| > |c_1| > 0 \\ |b_i| \geq |a_i| + |c_i| \quad (a_i c_i \neq 0) \\ |b_n| > |a_n| > 0 \end{cases} \quad (4.4.3)$$

称  $A$  为对角占优的三对角线矩阵。

可以验证, 式 (4.4.2) 中的  $A$  可以分解为二对角的下三角矩阵  $L$  和二对角的上三角矩阵  $U$  的乘积, 即  $A = LU$ 。当  $L$  的对角线元素为 1 时, 属于 Doolittle 分解; 当  $U$  的对角线元素为 1 时, 属于 Crout 分解。以下对  $A$  进行 Crout 分解:

令

$$\mathbf{L} = \begin{pmatrix} l_1 & & & & \\ a_2 & l_2 & & & \\ & a_3 & l_3 & & \\ & & \ddots & \ddots & \\ & & & a_n & l_n \end{pmatrix}, \mathbf{U} = \begin{pmatrix} 1 & u_1 & & & \\ & 1 & u_2 & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & u_{n-1} \\ & & & & & 1 \end{pmatrix} \quad (4.4.4)$$

对照式 (4.4.2) 中的  $\mathbf{A}$  和式 (4.4.4) 中的  $\mathbf{L}, \mathbf{U}$ ，利用矩阵的乘法，可以得

$$\begin{cases} b_1 = l_1 \\ c_i = l_i u_i \\ b_{i+1} = a_{i+1} u_i + l_{i+1} \end{cases} \quad (i = 1, 2, 3, \dots, n-1) \quad (4.4.5)$$

从式 (4.4.5) 解出

$$\begin{cases} l_1 = b_1 \\ u_i = c_i / l_i \\ l_{i+1} = b_{i+1} - a_{i+1} u_i \end{cases} \quad (i = 1, 2, 3, \dots, n-1) \quad (4.4.6)$$

可见，对  $\mathbf{A}$  的分解只需求  $l_i, u_i$ ，且按  $l_1 \rightarrow u_1 \rightarrow l_2 \rightarrow u_2 \rightarrow \dots \rightarrow l_{n-1} \rightarrow u_{n-1} \rightarrow l_n$  的递推过程进行，形象地称为“追”的过程。

这样，解方程组 (4.4.1) 就化为求  $\mathbf{L}\mathbf{U}\mathbf{x} = \mathbf{f}$ ，令  $\mathbf{U}\mathbf{x} = \mathbf{y}$ ，则  $\mathbf{L}\mathbf{y} = \mathbf{f}$ 。

解方程组  $\mathbf{L}\mathbf{y} = \mathbf{f}$ ，即

$$\begin{cases} l_1 y_1 = f_1 \\ a_i y_{i-1} + l_i y_i = f_i \end{cases} \quad (i = 2, \dots, n)$$

得

$$\begin{cases} y_1 = f_1 / l_1 \\ y_i = (f_i - a_i y_{i-1}) / l_i \end{cases} \quad (i = 2, \dots, n) \quad (4.4.7)$$

解方程组  $\mathbf{U}\mathbf{x} = \mathbf{y}$ ，即

$$\begin{cases} x_i + u_i x_{i+1} = y_i \\ x_n = y_n \end{cases} \quad (i = 1, 2, \dots, n)$$

得

$$\begin{cases} x_n = y_n \\ x_i = y_i - u_i x_{i+1} \end{cases} \quad (i = n-1, \dots, 2, 1) \quad (4.4.8)$$

形象地称回代求解过程，即式 (4.4.8) 为“赶”的过程，由式 (4.4.6)、式 (4.4.7) 和式 (4.4.8) 求解方程组 (4.4.1) 的方法称为追赶法。

不过，从计算  $y_i$  的式 (4.4.7) 可以看出，只要算出  $l_i$  和  $y_{i-1}$  就可以计算  $y_i$ ，所以可将计算式 (4.4.7) 归于“追”的过程，即求  $\mathbf{L}, \mathbf{U}, \mathbf{y}$  可以组织在一个循环中。如图 4.4.1 所示为追赶法的算法框图。

当系数矩阵  $\mathbf{A}$  满足式 (4.4.3) 时，用数学归纳法可以证明

$$\begin{aligned} 0 < |u_i| < 1 & \quad (i = 1, 2, \dots, n-1) \\ 0 < |b_i| - |a_i| \leq |b_i| + |a_i| & \quad (i = 1, 2, \dots, n) \end{aligned}$$



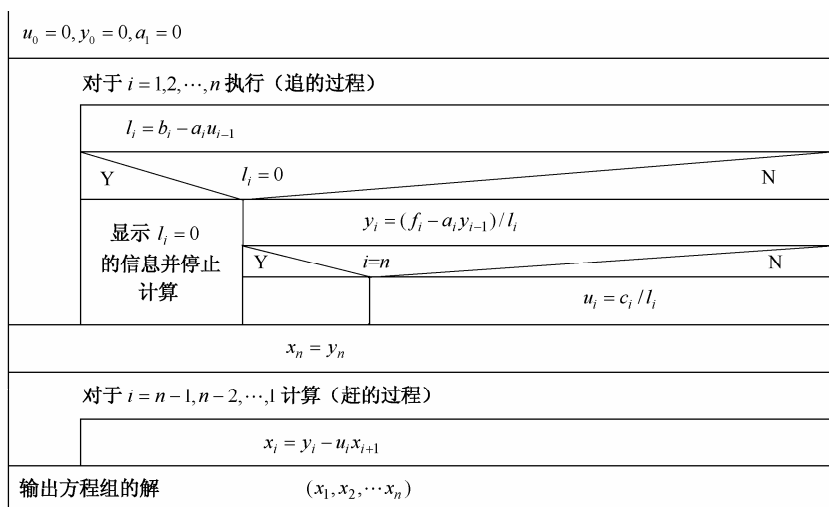


图 4.4.1 追赶法的算法框图

由于  $\{l_i\}, \{u_i\}$  有界, 且  $\{l_i\}$  在数量级上与  $A$  的元素相当, 因此, 即使不选主元, 追赶法在一般情况下也是数值稳定的。又因为  $\det(A) = \prod_{i=1}^n l_i \neq 0$  所以式 (4.4.1) 的解存在并且唯一。

追赶法的计算量为  $5n-4$  次乘、除法, 可用 4 个一维数组存放  $\{a_i\}, \{b_i\}, \{c_i\}, \{f_i\}$ , 共占用  $4n-2$  个单元。在计算过程中  $\{l_i\}, \{u_i\}, \{y_i\}$  依次覆盖掉  $\{b_i\}, \{c_i\}, \{f_i\}$ , 最后,  $\{x_i\}$  覆盖掉  $\{y_i\}$ , 所以, 追赶法具有计算量小, 占用内存单元少的特点。

例 4.4.1 用追赶法解方程组。

$$\begin{pmatrix} 3 & 1 & & \\ 2 & 3 & 1 & \\ & 2 & 3 & 1 \\ & & 1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

解 按式 (4.4.6) 和式 (4.4.7) 计算  $l_i, y_i, u_i$  的值, 结果见表 4.4.1。

表 4.4.1 例 4.4.1 计算结果

$i$	$l_i$	$u_i$	$y_i$
1	3	1/3	1/3
2	7/3	3/7	-2/7
3	15/7	7/15	11/15
4	38/15		-11/38

按式 (4.4.8) 计算  $x_i$  得

$$x = (21/38, -25/38, 33/38, -11/38)^T$$

例 4.4.2 解三对角方程组。

$$\begin{cases} 4x_1 + 2x_2 & = 1 \\ x_1 + 4x_2 + 2x_3 & = 2 \\ x_2 + 4x_3 + 2x_4 & = 3 \\ x_3 + 4x_4 & = 4 \end{cases}$$

解 利用如图 4.4.1 所示算法可得追的过程如下。

$$\text{第 1 步: } \begin{cases} l_1 = b_1 - a_1 u_0 = 4 \\ y_1 = (f_1 - a_1 y_0) / l_1 = 1/4 \\ u_1 = c_1 / l_1 = 1/2 \end{cases}$$

$$\text{第 2 步: } \begin{cases} l_2 = b_2 - a_2 u_1 = 4 - 1/2 = 7/2 \\ y_2 = (f_2 - a_2 y_1) / l_2 = (2 - 1/4) / 7/2 = 1/2 \\ u_2 = c_2 / l_2 = 4/7 \end{cases}$$

$$\text{第 3 步: } \begin{cases} l_3 = b_3 - a_3 u_2 = 24/7 \\ y_3 = (f_3 - a_3 y_2) / l_3 = 35/48 \\ u_3 = c_3 / l_3 = 7/12 \end{cases}$$

$$\text{第 4 步: } \begin{cases} l_4 = b_4 - a_4 u_3 = 41/12 \\ y_4 = (f_4 - a_4 y_3) / l_4 = 157/164 \end{cases}$$

赶的过程:  $x_4 = y_4 = 157/164$

$$x_3 = y_3 - u_3 x_4 = 7/41$$

$$x_2 = y_2 - u_2 x_3 = 33/82$$

$$x_1 = y_1 - u_1 x_2 = 2/41$$

## 4.5 误差分析

### 4.5.1 病态方程组与条件数

一个线性方程组  $Ax = b$  是由它的系数矩阵  $A$  和它的右端项  $b$  所确定, 在实际问题中, 由于各种原因,  $A$  或  $b$  往往有误差, 从而使得解也产生误差。本节讨论方程组的系数矩阵  $A$  或右端项  $b$  的微小误差对解向量的影响问题。

例 4.5.1 解方程组  $Ax = b$ 。

$$\begin{pmatrix} 2 & 6 \\ 2 & 6.0001 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 8 \\ 8.0001 \end{pmatrix}$$

解 其准确解为  $x^* = (1, 1)^T$ , 当  $A$  与  $b$  有微小变化时, 如  $\tilde{A} = A + \delta A$ ,  $\tilde{b} = b + \delta b$ , 则变为以下方程组

$$\begin{pmatrix} 2 & 6 \\ 2 & 5.9999 \end{pmatrix} \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{pmatrix} = \begin{pmatrix} 8 \\ 8.0002 \end{pmatrix}$$

准确解变为  $\tilde{x} = x + \delta x = (10, -2)^T$ 。

例 4.5.2 解方程组  $Ax = b$ 。

$$\begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix}$$

解 其准确解  $x^* = (1, 1, 1, 1)^T$ , 当  $A$  没有变化, 右端项有微小变化

$$b + \delta b = (32.1, 22.9, 33.1, 30.9)^T$$

则方程组的解变为

$$\mathbf{x} + \delta\mathbf{x} = (9.2, -12.6, 4.5, -1.1)^T$$

上述两个例子表明， $\mathbf{A}$  与  $\mathbf{b}$  的微小变化会引起方程组解  $\mathbf{x}$  的很大变化，所谓“差之毫厘，失之千里”。这种现象的出现是完全由方程组的性态决定的。

**定义 4.5.1 (病态方程组定义)** 求解线性方程组  $\mathbf{Ax} = \mathbf{b}$  时，若  $\mathbf{A}$  或  $\mathbf{b}$  有微小扰动  $\|\delta\mathbf{A}\|$  或  $\|\delta\mathbf{b}\|$  时，解  $\mathbf{x}$  的扰动  $\|\delta\mathbf{x}\|$  很大，则称此方程组为病态方程组，相应的系数矩阵  $\mathbf{A}$  称为病态矩阵。反之，若此时  $\|\delta\mathbf{x}\|$  很小，则称方程组为良态方程组，矩阵  $\mathbf{A}$  称为良态矩阵。

**定义 4.5.2** 称  $\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|}$ 、 $\frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|}$  和  $\frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|}$  分别为解向量  $\mathbf{x}$ 、矩阵  $\mathbf{A}$  和右端向量  $\mathbf{b}$  的相对扰动。

现在讨论方程组“病态”程度的标准。

设线性方程组

$$\mathbf{Ax} = \mathbf{b} \quad (4.5.1)$$

的扰动方程组为

$$(\mathbf{A} + \delta\mathbf{A})(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b} \quad (4.5.2)$$

式中， $\delta\mathbf{A}$  为  $\mathbf{A}$  的扰动矩阵， $\delta\mathbf{x}$  和  $\delta\mathbf{b}$  分别为  $\mathbf{x}$  和  $\mathbf{b}$  的扰动向量，并总假设  $\mathbf{A} + \delta\mathbf{A}$  非奇异。

为讨论方便，将扰动方程组分为以下两种情况。

(1)  $\delta\mathbf{A} = \mathbf{0}, \delta\mathbf{b} \neq \mathbf{0}, \mathbf{b} \neq \mathbf{0}$

此时扰动方程组为

$$\mathbf{A}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b} \quad (4.5.3)$$

式 (4.5.3) 和式 (4.5.1) 相减得

$$\mathbf{A}\delta\mathbf{x} = \delta\mathbf{b}$$

所以  $\delta\mathbf{x} = \mathbf{A}^{-1}\delta\mathbf{b}$ ，由范数的定义得

$$\|\delta\mathbf{x}\| \leq \|\mathbf{A}^{-1}\| \cdot \|\delta\mathbf{b}\|$$

再由  $\mathbf{Ax} = \mathbf{b}$  得

$$\|\mathbf{b}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{x}\|$$

从而有

$$\|\delta\mathbf{x}\| \cdot \|\mathbf{b}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\| \cdot \|\mathbf{x}\| \cdot \|\delta\mathbf{b}\|$$

因为  $\mathbf{b} \neq \mathbf{0}, \mathbf{x} \neq \mathbf{0}$ ，所以  $\|\mathbf{b}\| > 0, \|\mathbf{x}\| > 0$

于是有

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\| \cdot \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|} \quad (4.5.4)$$

式 (4.5.4) 说明当右端项  $\mathbf{b}$  有扰动  $\|\delta\mathbf{b}\|$  时，解  $\mathbf{x}$  相对扰动不超过  $\mathbf{b}$  的相对扰动的  $\|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$  倍。

(2)  $\delta\mathbf{A} \neq \mathbf{0}, \delta\mathbf{b} = \mathbf{0}, \mathbf{A} \neq \mathbf{0}$

此时扰动方程组为

$$(\mathbf{A} + \delta\mathbf{A})(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} \quad (4.5.5)$$

式 (4.5.5) 与式 (4.5.1) 相减得

$$\delta\mathbf{A}(\mathbf{x} + \delta\mathbf{x}) + \mathbf{A}\delta\mathbf{x} = \mathbf{0}$$

则

$$\delta \mathbf{x} = -\mathbf{A}^{-1} \delta \mathbf{A} (\mathbf{x} + \delta \mathbf{x})$$

由范数的定义得

$$\|\delta \mathbf{x}\| \leq \|\mathbf{A}^{-1}\| \cdot \|\delta \mathbf{A}\| \cdot \|\mathbf{x} + \delta \mathbf{x}\|$$

于是有

$$\frac{\|\delta \mathbf{x}\|}{\|\mathbf{x} + \delta \mathbf{x}\|} \leq \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\| \cdot \frac{\|\delta \mathbf{A}\|}{\|\mathbf{A}\|} \quad (4.5.6)$$

式 (4.5.6) 说明, 当  $\mathbf{A}$  有扰动  $\|\delta \mathbf{A}\|$  时, 解  $\mathbf{x} + \delta \mathbf{x}$  的相对扰动, 不超过  $\mathbf{A}$  的相对扰动的  $\|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$  倍。

当方程组 (4.5.1) 的系数矩阵  $\mathbf{A}$  和右端项  $\mathbf{b}$  同时有扰动  $\|\delta \mathbf{A}\|$  和  $\|\delta \mathbf{b}\|$  时, 在

$$\|\mathbf{A}^{-1} \cdot \delta \mathbf{A}\| \leq \|\mathbf{A}^{-1}\| \cdot \|\delta \mathbf{A}\| < 1$$

的条件下, 可推出

$$\frac{\|\delta \mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{\|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|}{1 - \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\| \cdot \frac{\|\delta \mathbf{A}\|}{\|\mathbf{A}\|}} \left( \frac{\|\delta \mathbf{b}\|}{\|\mathbf{b}\|} + \frac{\|\delta \mathbf{A}\|}{\|\mathbf{A}\|} \right) \quad (4.5.7)$$

由式 (4.5.4)、式 (4.5.6) 和式 (4.5.7) 可知,  $\|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$  刻画了方程组  $\mathbf{A}\mathbf{x} = \mathbf{b}$  的“病态”程度以及解对  $\mathbf{A}, \mathbf{b}$  扰动的敏感程度。

**定义 4.5.3** 设  $\mathbf{A}$  为  $n$  阶可逆矩阵, 称

$$\text{cond}(\mathbf{A}) = \|\mathbf{A}^{-1}\| \cdot \|\mathbf{A}\|$$

为矩阵  $\mathbf{A}$  的条件数, 式中  $\|\mathbf{A}\|$  是矩阵  $\mathbf{A}$  的算子范数。

常用的条件数为

$$\text{cond}_{\infty}(\mathbf{A}) = \|\mathbf{A}\|_{\infty} \cdot \|\mathbf{A}^{-1}\|_{\infty}$$

$$\text{cond}_1(\mathbf{A}) = \|\mathbf{A}\|_1 \cdot \|\mathbf{A}^{-1}\|_1$$

$$\text{cond}_2(\mathbf{A}) = \|\mathbf{A}\|_2 \cdot \|\mathbf{A}^{-1}\|_2$$

分别称为  $\mathbf{A}$  的  $\infty$ -条件数、1-条件数和 2-条件数。

条件数的性质:

① 对任意非奇异矩阵  $\mathbf{A}$ , 都有

$$\text{cond}(\mathbf{A}) \geq 1, \text{cond}(\mathbf{A}) = \text{cond}(\mathbf{A}^{-1})$$

② 若  $\mathbf{A}$  为非奇异阵, 且  $k \neq 0$  (常数), 则

$$\text{cond}(k\mathbf{A}) = \text{cond}(\mathbf{A})$$

③ 如果  $\mathbf{A}$  为非奇异阵,  $\mathbf{U}$  为正交矩阵, 则

$$\text{cond}_2(\mathbf{A}) = \text{cond}_2(\mathbf{UA}) = \text{cond}_2(\mathbf{AU}), \text{cond}_2(\mathbf{U}) = 1$$

④ 若  $\lambda_1, \lambda_n$  为  $\mathbf{A}$  的按模最大和最小的特征值, 则

$$\text{cond}_2(\mathbf{A}) \geq \frac{|\lambda_1|}{|\lambda_2|}$$

若  $\mathbf{A}$  对称, 则

$$\text{cond}_2(A) = \frac{|\lambda_1|}{|\lambda_2|}$$

若  $A$  对称正定, 则

$$\text{cond}_2(A) = \frac{\lambda_1}{\lambda_2}$$

**例 4.5.3** 设  $A, B \in R^{n \times n}$  为非奇异矩阵, 证明:

$$(1) \text{cond}(A) \geq 1, \text{cond}(A) = \text{cond}(A^{-1})$$

$$(2) \text{cond}(aA) = \text{cond}(A), \forall a \in R^1, a \neq 0$$

$$(3) \text{cond}(AB) \leq \text{cond}(A) \cdot \text{cond}(B)$$

$$\text{证明} \quad (1) \text{cond}(A) = \|A^{-1}\| \cdot \|A\| \geq \|A^{-1}A\| = \|I\| = 1$$

$$\text{cond}(A) = \|A^{-1}\| \cdot \|A\| = \|(A^{-1})^{-1}\| \cdot \|A^{-1}\| = \text{cond}(A^{-1})$$

$$(2) \text{cond}(aA) = \|(aA)^{-1}\| \cdot \|aA\| = \|A^{-1}\| \cdot \|A\| = \text{cond}(A)$$

$$(3) \text{cond}(AB) = \|(AB)^{-1}\| \cdot \|AB\| \leq \|A^{-1}\| \cdot \|B^{-1}\| \cdot \|B\| \cdot \|A\|$$

$$= \|A^{-1}\| \cdot \|A\| \cdot \|B^{-1}\| \cdot \|B\| = \text{cond}(A) \cdot \text{cond}(B)$$

矩阵的条件数是一个十分重要的概念, 当  $A$  的条件数相对较大时, 称式 (4.5.1) 是病态的,  $A$  称为“病态”矩阵; 当  $A$  的条件数相对较小时, 式 (4.5.1) 是“良态”的,  $A$  称为“良态”矩阵。用一个稳定的方法解一个良态的方程组时, 必然得到较精确的结果; 用一个稳定的方法解一个“病态”的方程组时, 结果也可能很不理想。

**例 4.5.4** 设在例 4.5.1 中的方程组  $Ax = b$  中,  $b$  有扰动  $\delta b = (0, 0.00001)^T$  试计算  $\text{cond}_\infty(A)$ , 并说明  $\delta b$  对解向量  $x$  的影响。

**解** 因为  $A = \begin{pmatrix} 2 & 6 \\ 2 & 6.0001 \end{pmatrix}$  非奇异, 所以  $A^{-1}$  存在。易得

$$A^{-1} = \begin{pmatrix} 30000.5 & -30000 \\ -100000 & 100000 \end{pmatrix}$$

则

$$\text{cond}_\infty(A) = \|A\|_\infty \times \|A^{-1}\|_\infty = 8.00001 \times 600000.5 \approx 4.8 \times 10^6$$

$$\frac{\|\delta x\|_\infty}{\|x\|_\infty} \leq \text{cond}_\infty(A) \frac{\|\delta b\|_\infty}{\|b\|_\infty} \approx 4.8 \times 10^6 \times \frac{0.00001}{8} \approx 6 = 600\%$$

这说明, 右端项  $b$  其分量的约 0.001% (0.00001) 的变化, 可能引起解向量  $x$  的 600% 的变化, 因此例 4.5.1 的方程组  $Ax = b$  是“病态”方程组,  $A$  为“病态”矩阵。

**例 4.5.5** 分析例 4.5.2 中方程组的“病态”程度。

**解** 因为例 4.5.2 中的系数矩阵  $A$  对称正定, 并可算出  $A$  的特征值如下

$$\lambda_1 \approx 30.2887, \lambda_2 \approx 3.858, \lambda_3 \approx 0.8431, \lambda_4 \approx 0.01015$$

所以

$$\text{cond}_2(A) = \frac{\lambda_1}{\lambda_4} \approx 2984$$

可见, 解的相对误差可能放大近 3000 倍, 故例 4.5.2 中的方程组是“病态”方程组。

## 4.5.2 病态方程组的解法

判断一个线性方程组  $Ax=b$  是否病态，需要计算系数矩阵的条件数  $\text{cond}(A)=\|A\|\cdot\|A^{-1}\|$ 。

但是，条件数的计算首先要计算逆矩阵的范数，计算量很大。在实际中，可通过求解过程直观地判断方程组的病态特征。

① 若在主元素消元过程中出现小主元，则  $A$  可能是病态阵，但病态阵未必一定有这种小主元。

② 系数矩阵的行列式的值相对来说很小，则  $A$  有可能是病态阵。

③ 从矩阵本身来看，若元素间数量级很大且无一定规律，或者矩阵的某些行（列）近似线性相关，这样的矩阵有可能是病态的。

④ 如果  $A$  的最大特征值和最小特征值之比（按绝对值）较大，则  $A$  有可能是病态的。

用选主元的消去法不能解决病态问题，对于病态方程组可采用以下措施：

① 采用高精度的运算，减轻病态影响。

② 采用预处理方法改善  $A$  的条件数，例如，可选择非奇异的对角阵或三角阵  $P$ 、 $Q$  将  $Ax=b$  转化为以下等价形式

$$\begin{cases} PAQy = Pb \\ y = Q^{-1}x \end{cases}$$

使  $\text{cond}(PAQ) < \text{cond}(A)$ 。

③ 当矩阵  $A$  的元素数量级较大时，对  $A$  的行（列）乘以适当的比例因子，使  $A$  的所有行列按  $\infty$ -范数大体上有相同的长度，使  $A$  的系数均衡，可使  $A$  的条件数得到改善。

**例 4.5.6** 计算线性方程组

$$\begin{pmatrix} 1 & 10^4 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 10^4 \\ 2 \end{pmatrix}$$

的条件数  $\text{cond}_{\infty}(A)$ 。

解

$$A = \begin{pmatrix} 1 & 10^4 \\ 1 & 1 \end{pmatrix}, \quad A^{-1} = \frac{1}{10^4 - 1} \begin{pmatrix} -1 & 10^4 \\ 1 & -1 \end{pmatrix}$$

$$\text{cond}_{\infty}(A) = \frac{(1+10^4)^2}{10^4 - 1} \approx 10^4$$

若对第 1 行引进比例因子  $s_1 = \max_{1 \leq i \leq 2} \|a_{1i}\| = 10^4$ ，第 1 行除以  $s_1$ ，得方程组  $A'x = b'$  即

$$\begin{pmatrix} 10^{-4} & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

而

$$(A')^{-1} = \frac{1}{1-10^{-4}} \begin{bmatrix} -1 & 1 \\ 1 & -10^{-4} \end{bmatrix}$$

于是  $\text{cond}_{\infty}(A') = \frac{4}{1-10^{-4}} \approx 4$ ，条件数有了很大的改善。

## 本章小结

本章介绍了适用于解中低阶稠密方程组的直接法。

高斯消去法是一种古典的解法，有消元和回代两个过程，其计算量远比克莱姆法则的计算量小。高斯-约当消去法只有消元过程而无回代过程，但计算量略比高斯消去法大。为保证计算过程顺利、稳定，每次消元都要选主元，一般采用列主元。三角分解法是消去法的变形，适用于求解系数矩阵相同的若干方程组。利用更简单的三角分解，可以得到求三对角方程组的追赶法。它们具有计算量小、占用内存少、方法简单、算法稳定的特点。如何避免舍入误差的增长是设计直接法时必须考虑的问题，误差的影响取决于方程组的条件数  $\text{cond}(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$ ，条件数大时，称为病态方程组，对病态方程组必须采取慎重措施，才能求出比较准确的解。

### 习题 4

4.1 用高斯消去法解方程组 
$$\begin{cases} 2x_1 + x_2 + x_3 = 4 \\ x_1 + 3x_2 + 2x_3 = 6 \\ x_1 + 2x_2 + 2x_3 = 5 \end{cases}。$$

4.2 用高斯消去法解方程组 
$$\begin{cases} x_1 + x_2 + 3x_4 = 4 \\ 2x_1 + x_2 - x_3 + x_4 = 1 \\ 3x_1 - x_2 - x_3 + 2x_4 = -3 \\ -x_1 + 2x_2 + 3x_3 - x_4 = 4 \end{cases}。$$

4.3 用列主元素高斯消去法解线性方程组。

$$\begin{pmatrix} 0 & 2 & 0 & 1 \\ 2 & 2 & 3 & 2 \\ 4 & -3 & 0 & 1 \\ 6 & 1 & -6 & -5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 \\ -2 \\ -7 \\ 6 \end{pmatrix}$$

4.4 用列主元素高斯消去法求矩阵的行列式。

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 2 & 1 \\ 2 & 3 & 5 & 7 \\ -1 & 6 & 2 & 4 \\ 5 & 1 & 9 & 6 \end{pmatrix}$$

4.5 用列主元素高斯消去法解方程组。

$$\begin{pmatrix} -3 & 2 & 6 \\ 10 & -7 & 0 \\ 5 & -1 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 4 \\ 7 \\ 6 \end{pmatrix}$$

4.6 设  $\mathbf{A} = (a_{ij})$ ,  $a_{11} \neq 0$ , 经过一步高斯消去得到  $\mathbf{A}^{(2)} = \begin{pmatrix} a_{11} & a_1^T \\ & \mathbf{A}_2 \end{pmatrix}$ , 式中

$$A_2 = \begin{pmatrix} a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \ddots & \vdots \\ a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{pmatrix}$$

求证：(1) 若  $A$  对称，则  $A_2$  也对称。

(2) 若  $A$  严格对角占优，则  $A_2$  也严格对角占优。

4.7 用追赶法解三对角线方程组 
$$\begin{cases} 2x_1 + x_2 = 3 \\ x_1 + 2x_2 - 3x_3 = -3 \\ 3x_2 - 7x_3 + 4x_4 = -10 \\ 2x_3 + 5x_4 = 2 \end{cases}。$$

4.8 用追赶法解三对角线方程组  $Ax = b$ ，式中  $A = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix}$ ， $b = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ 。

4.9 用追赶法解三对角方程组 
$$\begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 3 & -2 & 0 \\ 0 & -2 & 4 & -2 \\ 0 & 0 & -3 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 6 \\ 1 \\ 0 \\ 1 \end{pmatrix}。$$

4.10 用直接分解法求解方程组 
$$\begin{cases} 2x_1 + x_2 + x_3 = 4 \\ x_1 + 3x_2 + 2x_3 = 6 \\ x_1 + 2x_2 + 2x_3 = 5 \end{cases}。$$

4.11 设  $L$  是非奇异下三角形矩阵，要求：(1) 列出逐次代入求解  $Lx = f$  的公式。(2) 上述求解过程需要多少次乘、除法运算？(3) 给出求  $L^{-1}$  的计算公式。

4.12 设  $A = \begin{pmatrix} 1 & 1 \\ -5 & 1 \end{pmatrix}$ ，则  $A$  的谱半径  $\rho(A)$  为多少？ $A$  的条件数  $\text{cond}(A)_\infty$  为多少？

4.13 设  $A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$ ，则  $\text{Cond}(A)_2$  为多少？

4.14 设  $A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & a \\ 0 & a & 2 \end{pmatrix}$ ，为使  $A$  可分解为  $A = LL^T$ ，式中  $L$  为对角线元素为正的下三角形

矩阵，求  $a$  的取值范围。若取  $a=1$ ，则写出矩阵  $L$ 。



## 第5章 解线性方程组的迭代法



### 学习要点

迭代法是用某种极限过程去逐步逼近线性方程组精确解的方法。本章介绍几种常见的迭代法，主要内容有：

- (1) 迭代公式的构造。
- (2) 迭代方法，包括雅可比迭代法、高斯-塞德尔迭代法。
- (3) 迭代法的收敛性判断：
  - ① 基本收敛定理：迭代矩阵的谱半径  $\rho(B) < 1$ 。
  - ② 充分条件：迭代矩阵的范数  $\|B\| < 1$ ，系数矩阵为严格对角占优矩阵。



### 教学建议

在本章介绍的迭代法中，雅可比迭代法和高斯-塞德尔迭代法是重点，要求学生掌握这两种迭代公式的构造和收敛性判断，并能求解一般线性方程组的解，了解迭代法的收敛性判断定理。建议学时为 4~6 学时。

## 5.1 引言

第4章介绍的解线性方程组的直接法是解低阶稠密方程组的有效方法。但是，在工程技术中常产生大型高阶稀疏矩阵方程组，例如由某些偏微分方程数值解所产生的线性方程组  $Ax = b$ ,  $A$  的阶数很大 ( $n \geq 10^4$ )，但零元素较多。迭代法是能够充分利用系数矩阵稀疏性特点来求线性方程组的有效方法。迭代法的基本思想是用逐次逼近的方法求线性方程组的解。

设有方程组

$$Ax = b \quad (5.1.1)$$

将其转化为等价的便于迭代的形式

$$x = Bx + f \quad (5.1.2)$$

(这种转化总能实现，如令  $B = I - A$ ,  $f = b$ ) 并由此构造迭代公式

$$x^{(k+1)} = Bx^{(k)} + f \quad (5.1.3)$$

其中， $B$  称为迭代矩阵， $f$  称为迭代向量。对任意的初始向量  $x^{(0)}$ ，由式 (5.1.3) 可求得向量序列  $\{x^{(k)}\}_0^\infty$ 。若  $\lim_{k \rightarrow \infty} x^{(k)} = x^*$ ，则  $x^*$  就是方程组 (5.1.1) 或方程组 (5.1.2) 的解。此时称迭代公式 (5.1.3) 是收敛的，否则称为发散的 (见 2.3.11 节)。构造的迭代公式 (5.1.3) 是否收敛，取决于迭代矩阵  $B$  的性质。

例 5.1.1 用迭代法解方程组。

$$\begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 5 \\ 5 \end{pmatrix} \quad (5.1.4)$$

解 方程组 (5.1.4) 的精确解为  $x_1=1, x_2=2$ ，用迭代法解方程组，可由第一个方程求出  $x_1$ ，由第二个方程求出  $x_2$ ，从而将方程组转化为

$$\begin{cases} x_1 = -\frac{1}{3}x_2 + \frac{5}{3} \\ x_2 = -\frac{1}{2}x_1 + \frac{5}{2} \end{cases}$$

取初始向量  $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)})^T = (0, 0)^T$ ，构造迭代公式

$$\begin{cases} x_1^{(k+1)} = -\frac{1}{3}x_2^{(k)} + \frac{5}{3} \\ x_2^{(k+1)} = -\frac{1}{2}x_1^{(k)} + \frac{5}{2} \end{cases}$$

也可表示为矩阵形式

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^{(k+1)} = \begin{pmatrix} 0 & -\frac{1}{3} \\ -\frac{1}{2} & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^{(k)} + \begin{pmatrix} \frac{5}{3} \\ \frac{5}{2} \end{pmatrix} \quad (5.1.5)$$

计算结果见表 5.1.1。

表 5.1.1 例 5.1.1 的计算结果 1

$k$	0	1	2	...	9	10	...
$x_1^{(k)}$	0	1.6667	0.8333	...	1.0005	0.9998	...
$x_2^{(k)}$	0	2.5	1.6667	...	2.0004	1.9997	...

从表 5.1.1 可以看出，由式 (5.1.5) 计算出的  $\mathbf{x}^{(k)} = (x_1^{(k)}, x_2^{(k)})^T$ ，收敛于方程组 (5.1.4) 的精确解  $\mathbf{x}^* = (1, 2)^T$ ，因此，迭代公式 (5.1.5) 收敛。可取  $\mathbf{x}^{(10)} = (0.9998, 1.9997)^T$  作为精确解  $\mathbf{x}^* = (1, 2)^T$  的近似值。

但是，若由第一个方程求出  $x_2$ ，由第二个方程求出  $x_1$ ，将方程组转化为

$$\begin{cases} x_1 = -2x_2 + 5 \\ x_2 = -3x_1 + 5 \end{cases}$$

仍取  $\mathbf{x}^{(0)} = (0, 0)^T$  构造迭代公式

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^{(k+1)} = \begin{pmatrix} 0 & -2 \\ -3 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^{(k)} + \begin{pmatrix} 5 \\ 5 \end{pmatrix} \quad (5.1.6)$$

则计算结果见表 5.1.2。

表 5.1.2 例 5.1.1 的计算结果 2

$k$	0	1	2	3	4	5	...
$x_1^{(k)}$	0	5	-5	25	-35	145	...
$x_2^{(k)}$	0	5	-10	20	-70	110	...

由表 5.1.2 可以看出, 显然由式 (5.1.6) 计算出的  $\{x^{(k)}\}$  发散, 因此迭代公式 (5.1.6) 发散。

## 5.2 雅可比迭代法

设有线性方程组

$$\sum_{j=1}^n a_{ij}x_j = b_i \quad (i=1,2,\cdots,n) \quad (5.2.1)$$

其矩阵形式为  $Ax=b$ , 设系数矩阵  $A$  为非奇异矩阵, 且  $a_{ii} \neq 0$  ( $i=1,2,\cdots,n$ ), 从式 (5.2.1) 的第  $i$  个方程中解出  $x_i$ , 得其等价形式

$$x_i = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1, j \neq i}^n a_{ij}x_j \right) \quad (i=1,2,\cdots,n) \quad (5.2.2)$$

取初始向量  $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \cdots, x_n^{(0)})^T$ , 对式 (5.2.2) 应用迭代法, 可建立相应的迭代公式

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( - \sum_{j=1, j \neq i}^n a_{ij}x_j^{(k)} + b_i \right) \quad (5.2.3)$$

也可记为矩阵形式

$$x^{(k+1)} = B_J x^{(k)} + f_J \quad (5.2.4)$$

若将系数矩阵  $A$  分解为  $A = D - L - U$ , 式中

$$D = \begin{pmatrix} a_{11} & & & & \\ & a_{22} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & a_{nn} \end{pmatrix}$$

$$-L = \begin{pmatrix} 0 & & & & \\ a_{21} & 0 & & & \\ a_{31} & a_{32} & 0 & & \\ \vdots & \vdots & \ddots & \ddots & \\ a_{n1} & a_{n2} & \cdots & a_{nn-1} & 0 \end{pmatrix}$$

$$-U = \begin{pmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ & 0 & a_{23} & \cdots & a_{2n} \\ & & 0 & \ddots & \vdots \\ & & & \ddots & a_{n-1n} \\ & & & & 0 \end{pmatrix}$$

则方程组  $Ax=b$  变为

$$(D - L - U)x = b$$

得

$$Dx = (L + U)x + b$$

于是

$$\begin{aligned} x &= D^{-1}(L + U)x + D^{-1}b = D^{-1}(D - A)x + D^{-1}b \\ &= (I - D^{-1}A)x + D^{-1}b = B_J x + f_J \end{aligned}$$

于是式 (5.2.4) 中的  $B_J = I - D^{-1}A$ ,  $f_J = D^{-1}b$ 。式 (5.2.3) 和式 (5.2.4) 分别称为雅可比 (Jacobi) 迭代法的分量形式和矩阵形式, 分量形式用于编程计算, 矩阵形式用于讨论迭代法的收敛性。

**例 5.2.1** 用雅可比迭代法求解线性方程组。

$$\begin{cases} 10x_1 - x_2 - 2x_3 = 72 \\ -x_1 + 10x_2 - 2x_3 = 83 \\ -x_1 - x_2 + 5x_3 = 42 \end{cases}$$

**解** 分别从 3 个方程中求出  $x_1, x_2, x_3$ , 得雅可比迭代公式

$$\begin{cases} x_1^{(k+1)} = \frac{1}{10}(x_2^{(k)} + 2x_3^{(k)} + 72) \\ x_2^{(k+1)} = \frac{1}{10}(x_1^{(k)} + 2x_3^{(k)} + 83) \\ x_3^{(k+1)} = \frac{1}{5}(x_1^{(k)} + x_2^{(k)} + 42) \end{cases}$$

其矩阵形式为

$$\mathbf{x}^{(k+1)} = \begin{pmatrix} 0 & 0.1 & 0.2 \\ 0.1 & 0 & 0.2 \\ 0.2 & 0.2 & 0 \end{pmatrix} \mathbf{x}^{(k)} + \begin{pmatrix} 7.2 \\ 8.3 \\ 8.4 \end{pmatrix}$$

取  $\mathbf{x}^{(0)} = (0, 0, 0)^T$ , 代入上述迭代公式, 迭代结果见表 5.2.1。

表 5.2.1 例 5.2.1 的计算结果

$k$	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$k$	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
0	0.0000	0.0000	0.0000	5	10.9510	11.9510	12.9414
1	7.2000	8.3000	8.4000	6	10.9834	11.9834	12.9504
2	9.7100	10.7000	11.5000	7	10.9944	11.9981	12.9934
3	10.5700	11.5700	12.4820	8	10.9981	11.9941	12.9978
4	10.8525	11.8534	12.8282	9	10.9994	11.9994	12.9992

迭代 9 次, 得近似解  $\mathbf{x}^{(9)} = (10.9994, 11.9994, 12.9992)^T$ 。事实上, 此方程组的准确解为  $\mathbf{x} = (11, 12, 13)^T$ 。从表 5.2.1 可以看出, 随着迭代次数的增加, 迭代结果越来越接近精确解。

### 5.3 高斯-塞德尔迭代法

雅可比迭代法的优点是公式简单, 迭代矩阵容易计算。在每步迭代时, 用  $\mathbf{x}^{(k)}$  的全部分量代入求出  $\mathbf{x}^{(k+1)}$  的全部分量, 因此称为同步迭代法, 计算时需保留两个近似解向量  $\mathbf{x}^{(k)}$  和  $\mathbf{x}^{(k+1)}$ 。

但在雅可比迭代过程中, 对已经计算出的信息未能充分利用, 即在计算第  $i$  个分量  $x_i^{(k+1)}$  时, 已经计算出的最新分量  $x_1^{(k+1)}, \dots, x_{i-1}^{(k+1)}$  没有被利用。从直观上看, 在收敛的前提下, 这些新的分量  $x_1^{(k+1)}, \dots, x_{i-1}^{(k+1)}$  应该比旧分量  $x_1^{(k)}, \dots, x_{i-1}^{(k)}$  更好, 更精确一些。因此, 如果每计算出一个新的分量便立即用它取代对应的旧分量进行迭代, 可能收敛更快, 并且只需要存储一个近似解向量即可。根据此思想可构造出高斯-塞德尔 (Gauss-Seidel) 迭代法, 其迭代公式为

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( -\sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)} + b_i \right) \quad (i=1, 2, \dots, n) \quad (5.3.1)$$

也可以写成矩阵形式

$$\mathbf{x}^{(k+1)} = \mathbf{B}_{G-S} \mathbf{x}^{(k)} + \mathbf{f}_{G-S} \quad (5.3.2)$$

仍将系数矩阵  $\mathbf{A}$  分解为  $\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{U}$  ( $\mathbf{D}, \mathbf{L}, \mathbf{U}$  的含义与前面同), 则方程组变为

$$(\mathbf{D} - \mathbf{L} - \mathbf{U})\mathbf{x} = \mathbf{b}$$

得

$$\mathbf{D}\mathbf{x} = \mathbf{L}\mathbf{x} + \mathbf{U}\mathbf{x} + \mathbf{b}$$

将最新分量代替旧分量, 得

$$\mathbf{D}\mathbf{x}^{(k+1)} = \mathbf{L}\mathbf{x}^{(k+1)} + \mathbf{U}\mathbf{x}^{(k)} + \mathbf{b}$$

即

$$(\mathbf{D} - \mathbf{L})\mathbf{x}^{(k+1)} = \mathbf{U}\mathbf{x}^{(k)} + \mathbf{b}$$

于是有

$$\mathbf{x}^{(k+1)} = (\mathbf{D} - \mathbf{L})^{-1} \mathbf{U}\mathbf{x}^{(k)} + (\mathbf{D} - \mathbf{L})^{-1} \mathbf{b} = \mathbf{B}_{G-S} \mathbf{x}^{(k)} + \mathbf{f}_{G-S}$$

所以式 (5.3.2) 中的  $\mathbf{B}_{G-S} = (\mathbf{D} - \mathbf{L})^{-1} \mathbf{U}$ ,  $\mathbf{f}_{G-S} = (\mathbf{D} - \mathbf{L})^{-1} \mathbf{b}$ 。

**例 5.3.1** 用高斯-塞德尔迭代法求解例 5.2.1 的线性方程组。

**解** 高斯-塞德尔迭代公式为

$$\begin{cases} x_1^{(k+1)} = \frac{1}{10}(x_2^{(k)} + 2x_3^{(k)} + 72) \\ x_2^{(k+1)} = \frac{1}{10}(x_1^{(k+1)} + 2x_3^{(k)} + 83) \\ x_3^{(k+1)} = \frac{1}{5}(x_1^{(k+1)} + x_2^{(k+1)} + 42) \end{cases} \quad (5.3.3)$$

仍取  $\mathbf{x}^{(0)} = (0, 0, 0)^T$ , 计算结果见表 5.3.1。

表 5.3.1 例 5.3.1 的计算结果

$k$	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$k$	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
0	0.0000	0.0000	0.0000	4	10.9913	11.9947	12.9972
1	7.2000	9.0200	11.6440	5	10.9989	11.9993	12.9996
2	10.4308	11.6719	12.8205	6	10.9999	11.9999	13.0000
3	10.9313	11.9572	12.9778				

从表 5.2.1 与表 5.3.1 的比较可以看出, 用高斯-塞德尔迭代法比雅可比迭代法收敛快。这个结论在都收敛的前提下是成立的, 但也有相反的情况, 即雅可比迭代法收敛, 高斯-塞德尔迭代法发散的情形 (参见例 5.4.2 和例 5.4.3)。

在例 5.3.1 中, 迭代矩阵  $\mathbf{B}_{G-S}$  可以用两种方法求得。

方法 1: 利用  $\mathbf{B}_{G-S} = (\mathbf{D} - \mathbf{L})^{-1} \mathbf{U}$  计算, 其中

$$\mathbf{D}-\mathbf{L}=\begin{pmatrix} 10 & & \\ -1 & 10 & \\ -1 & -1 & 5 \end{pmatrix}, \quad (\mathbf{D}-\mathbf{L})^{-1}=\begin{pmatrix} \frac{1}{10} & & \\ \frac{1}{100} & \frac{1}{10} & \\ \frac{11}{500} & \frac{1}{50} & \frac{1}{5} \end{pmatrix}$$

所以

$$\mathbf{B}_{\text{G-S}}=(\mathbf{D}-\mathbf{L})^{-1}\mathbf{U}=\begin{pmatrix} \frac{1}{10} & & \\ \frac{1}{100} & \frac{1}{10} & \\ \frac{11}{500} & \frac{1}{50} & \frac{1}{5} \end{pmatrix}\begin{pmatrix} 0 & 1 & 2 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{pmatrix}=\begin{pmatrix} 0 & \frac{1}{10} & \frac{2}{10} \\ 0 & \frac{1}{100} & \frac{22}{100} \\ 0 & \frac{11}{500} & \frac{42}{500} \end{pmatrix}$$

方法 2: 将式 (5.3.3) 中的第一式代入第二式中, 将第二式右端的上标都化为  $k$  (暂不考虑常数项), 得

$$x_2^{(k+1)}=\frac{1}{10}\left(x_1^{(k+1)}+2x_3^{(k)}\right)=\frac{1}{10}\left(\frac{1}{10}\left(x_2^{(k)}+2x_3^{(k)}\right)+2x_3^{(k)}\right)=\frac{1}{100}x_2^{(k)}+\frac{22}{100}x_3^{(k)}$$

同理, 将式 (5.3.3) 中的第一、二式代入第三式中, 将第三式右端的上标都化为  $k$  (暂不考虑常数项), 得

$$x_3^{(k+1)}=\frac{1}{5}\left(x_1^{(k+1)}+x_2^{(k+1)}\right)=\frac{1}{5}\left(\frac{1}{10}\left(x_2^{(k)}+2x_3^{(k)}\right)+\frac{1}{100}x_2^{(k)}+\frac{22}{100}x_3^{(k)}\right)=\frac{11}{500}x_2^{(k)}+\frac{42}{500}x_3^{(k)}$$

写成矩阵形式 (不考虑常数项) 为

$$\mathbf{x}^{(k+1)}=\begin{pmatrix} 0 & \frac{1}{10} & \frac{2}{10} \\ 0 & \frac{1}{100} & \frac{22}{100} \\ 0 & \frac{11}{500} & \frac{42}{500} \end{pmatrix}\mathbf{x}^{(k)}$$

所以, 高斯-塞德尔迭代法的迭代矩阵为

$$\mathbf{B}_{\text{G-S}}=\begin{pmatrix} 0 & \frac{1}{10} & \frac{2}{10} \\ 0 & \frac{1}{100} & \frac{22}{100} \\ 0 & \frac{11}{500} & \frac{42}{500} \end{pmatrix}$$

## 5.4 迭代法的收敛性

**定理 5.4.1 (迭代法收敛性基本定理)** 设有  $n$  阶方程组  $\mathbf{x}=\mathbf{B}\mathbf{x}+\mathbf{f}$ , 对于任意初始向量  $\mathbf{x}^{(0)}$  和右端项  $\mathbf{f}$ , 迭代法收敛的充分必要条件是迭代矩阵的谱半径  $\rho(\mathbf{B})<1$ 。

**证明** 因为  $\rho(\mathbf{B})<1$ , 即  $\mathbf{B}$  的特征值  $|\lambda_i|<1$  ( $i=1,2,\cdots,n$ ), 矩阵  $\mathbf{I}-\mathbf{B}$  的特征值为

$$\mu_i=1-\lambda_i\neq 0 \quad (i=1,2,\cdots,n)$$

所以

$$\det(\mathbf{I} - \mathbf{B}) = \prod_{i=1}^n (1 - \lambda_i) \neq 0$$

即  $\mathbf{I} - \mathbf{B}$  非奇异, 因此, 方程组  $(\mathbf{I} - \mathbf{B})\mathbf{x} = \mathbf{f}$ , 即

$$\mathbf{x} = \mathbf{B}\mathbf{x} + \mathbf{f}$$

有唯一的解  $\mathbf{x}^*$ 。

令误差向量  $\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}^*$ , 则

$$\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}^* = (\mathbf{B}\mathbf{x}^{(k-1)} + \mathbf{f}) - (\mathbf{B}\mathbf{x}^* + \mathbf{f}) = \mathbf{B}(\mathbf{x}^{(k-1)} - \mathbf{x}^*) = \mathbf{B}\mathbf{e}^{(k-1)}$$

逐次递推得

$$\mathbf{e}^{(k)} = \mathbf{B}^k \mathbf{e}^{(0)} \quad (5.4.1)$$

先证明充分性: 因为  $\rho(\mathbf{B}) < 1$ , 则由定理 2.3.11, 得

$$\lim_{k \rightarrow \infty} \mathbf{B}^k = 0$$

由式 (5.4.1) 对任意的初值  $\mathbf{x}^{(0)}$  和迭代向量  $\mathbf{f}$  有

$$\lim_{k \rightarrow \infty} \mathbf{e}^{(k)} = 0$$

即

$$\lim_{k \rightarrow \infty} \mathbf{x}^{(k)} = \mathbf{x}^*$$

再证明必要性: 设对任意初始向量  $\mathbf{x}^{(0)}$  和迭代向量  $\mathbf{f}$  均有

$$\lim_{k \rightarrow \infty} \mathbf{x}^{(k)} = \mathbf{x}^*$$

从而

$$\mathbf{x}^* = \mathbf{B}\mathbf{x}^* + \mathbf{f}$$

由式 (5.4.1) 可得

$$\mathbf{x}^{(k)} - \mathbf{x}^* = \mathbf{B}^k (\mathbf{x}^{(0)} - \mathbf{x}^*)$$

对任意的初值  $\mathbf{x}^{(0)}$ , 有

$$\lim_{k \rightarrow \infty} \mathbf{B}^k (\mathbf{x}^{(0)} - \mathbf{x}^*) = 0$$

可推出

$$\lim_{k \rightarrow \infty} \mathbf{B}^k = 0$$

由定理 2.3.11 得

$$\rho(\mathbf{B}) < 1$$

定理得证。

该定理说明, 迭代法的收敛性取决于迭代矩阵  $\mathbf{B}$  的谱半径,  $\mathbf{B}$  又依赖于方程组的系数矩阵  $\mathbf{A}$ , 而与初始向量的选取和方程组的右端项无关。另外,  $\rho(\mathbf{B})$  越小, 序列  $\{\mathbf{x}^{(k)}\}$  收敛也越快, 由此可给出收敛速度的定义。

定义 5.4.1 称

$$R(\mathbf{B}) = -\ln \rho(\mathbf{B})$$

为迭代法的收敛速度。

**例 5.4.1** 试讨论例 5.1.1 的收敛性。

**解:** 因为迭代公式 (5.1.5) 中迭代矩阵

$$\mathbf{B} = \begin{pmatrix} 0 & -\frac{1}{3} \\ -\frac{1}{2} & 0 \end{pmatrix}$$

的特征值为:  $\lambda_1 = -\frac{1}{\sqrt{6}}$ ,  $\lambda_2 = \frac{1}{\sqrt{6}}$ , 所以

$$\rho(\mathbf{B}) = \frac{1}{\sqrt{6}} < 1$$

因此, 迭代公式 (5.1.5) 收敛。

因为迭代公式 (5.1.6) 中迭代矩阵

$$\mathbf{B} = \begin{pmatrix} 0 & -2 \\ -3 & 0 \end{pmatrix}$$

的特征值为:  $\lambda_1 = -\sqrt{6}$ ,  $\lambda_2 = \sqrt{6}$ , 所以

$$\rho(\mathbf{B}) = \sqrt{6} > 1$$

因此, 迭代公式 (5.1.6) 发散。

**例 5.4.2** 已知方程组  $\begin{pmatrix} 2 & -1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & -2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ , 试讨论分别用雅可比迭代法和高斯-塞德尔迭代法求解此方程组时的收敛性。

**解** 由构造雅可比迭代法和高斯-塞德尔迭代法系数矩阵的分解形式  $\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{U}$ , 可得

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & -2 \end{pmatrix} = \mathbf{D} - \mathbf{L} - \mathbf{U} = \begin{pmatrix} 2 & & \\ & 1 & \\ & & -2 \end{pmatrix} - \begin{pmatrix} 0 & & \\ -1 & 0 & \\ -1 & -1 & 0 \end{pmatrix} - \begin{pmatrix} 0 & 1 & -1 \\ & 0 & -1 \\ & & 0 \end{pmatrix}$$

用雅可比迭代法的迭代矩阵为

$$\mathbf{B}_J = \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U}) = \begin{pmatrix} \frac{1}{2} & & \\ & 1 & \\ & & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & -1 \\ -1 & -1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{2} & -\frac{1}{2} \\ -1 & 0 & -1 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$$

$$\text{其特征方程为 } |\lambda \mathbf{I} - \mathbf{B}_J| = \begin{vmatrix} \lambda & -\frac{1}{2} & \frac{1}{2} \\ 1 & \lambda & 1 \\ -\frac{1}{2} & -\frac{1}{2} & \lambda \end{vmatrix} = \lambda^3 + \frac{5}{4}\lambda = 0$$

解得  $\lambda_1 = 0$ ,  $\lambda_{2,3} = \pm \frac{\sqrt{5}}{2}i$ 。

由于  $\rho(\mathbf{B}_J) = \sqrt{5}/2 > 1$ , 故雅可比迭代法发散。

对高斯-塞德尔迭代法, 其迭代矩阵为



$$\mathbf{B}_G = (\mathbf{D} - \mathbf{L})^{-1} \mathbf{U} = \begin{pmatrix} 2 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & -2 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 1 & -1 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{2} & -\frac{1}{2} \\ 0 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & -\frac{1}{2} \end{pmatrix}$$

显然其特征值为  $\lambda_1 = 0, \lambda_{2,3} = -\frac{1}{2}, \rho(\mathbf{B}_G) = \frac{1}{2} < 1$ , 故高斯-塞德尔迭代法收敛。

**例 5.4.3** 已知方程组  $\begin{pmatrix} 1 & 2 & -2 \\ 1 & 1 & 1 \\ 2 & 2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ , 试讨论分别用雅可比迭代法和高斯-塞德尔迭代法求解此方程组时的收敛性。

**解** 由构造雅可比迭代法和高斯-塞德尔迭代法系数矩阵的分解形式  $\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{U}$ , 可得

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & -2 \\ 1 & 1 & 1 \\ 2 & 2 & 1 \end{pmatrix} = \mathbf{D} - \mathbf{L} - \mathbf{U} = \begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix} - \begin{pmatrix} 0 & & \\ -1 & 0 & \\ -2 & -2 & 0 \end{pmatrix} - \begin{pmatrix} 0 & -2 & 2 \\ & 0 & -1 \\ & & 0 \end{pmatrix}$$

则雅可比迭代法的迭代矩阵为

$$\mathbf{B}_J = \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U}) = \begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix} \begin{pmatrix} 0 & -2 & 2 \\ -1 & 0 & -1 \\ -2 & -2 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -2 & 2 \\ -1 & 0 & -1 \\ -2 & -2 & 0 \end{pmatrix}$$

其特征方程为  $|\lambda \mathbf{I} - \mathbf{B}_J| = \begin{vmatrix} \lambda & 2 & -2 \\ 1 & \lambda & 1 \\ 2 & 2 & \lambda \end{vmatrix} = \lambda^3 = 0$

解得  $\lambda_1 = \lambda_2 = \lambda_3 = 0$ , 由于  $\rho(\mathbf{B}_J) = 0 < 1$ , 故雅可比迭代法收敛。

对高斯-塞德尔迭代法, 其迭代矩阵为

$$\mathbf{B}_G = (\mathbf{D} - \mathbf{L})^{-1} \mathbf{U} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 2 & 2 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 0 & -2 & 2 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -2 & 2 \\ 0 & 2 & -3 \\ 0 & 0 & 2 \end{pmatrix}$$

显然  $\mathbf{B}_G$  的特征值为  $\lambda_1 = 0, \lambda_{2,3} = 2$ , 即  $\rho(\mathbf{B}_G) = 2$ , 故高斯-塞德尔迭代法发散。

**定理 5.4.2** 设雅可比迭代矩阵  $\mathbf{B}_J = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$  为非负矩阵, 则下列关系有且仅有一个成立

- (1)  $\rho(\mathbf{B}_J) = \rho(\mathbf{B}_{G-S}) = 0$
- (2)  $0 < \rho(\mathbf{B}_J) < \rho(\mathbf{B}_{G-S}) < 1$
- (3)  $\rho(\mathbf{B}_J) = \rho(\mathbf{B}_{G-S}) = 1$
- (4)  $1 < \rho(\mathbf{B}_J) < \rho(\mathbf{B}_{G-S})$

**定理 5.4.2** 说明当雅可比迭代矩阵  $\mathbf{B}_J = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$  为非负矩阵时, 雅可比方法和高斯-塞德尔方法同时收敛, 或者同时发散。若同时收敛, 则高斯-塞德尔方法比雅可比方法收敛得快。

一般当  $n$  较大时, 迭代矩阵  $\mathbf{B}$  的特征值计算比较复杂, 定理 5.4.1 的条件较难验证。利用定理 2.3.11 的结论  $\rho(\mathbf{B}) \leq \|\mathbf{B}\|$ , 用  $\|\mathbf{B}\|$  作为  $\rho(\mathbf{B})$  的上界的一种估计, 于是可得如下定理。

**定理 5.4.3** (迭代法收敛的充分条件 1) 若  $\|\mathbf{B}\| < 1$ , 则由迭代公式 (5.1.3) 所产生的向量序列  $\{x^{(k)}\}$  收敛于方程组  $\mathbf{x} = \mathbf{B}\mathbf{x} + \mathbf{f}$  的精确解  $\mathbf{x}^*$ , 且有误差估计式

$$\|\mathbf{x}^{(k)} - \mathbf{x}^*\| \leq \frac{\|\mathbf{B}\|}{1 - \|\mathbf{B}\|} \|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\| \quad (5.4.2)$$

$$\|\mathbf{x}^{(k)} - \mathbf{x}^*\| \leq \frac{\|\mathbf{B}\|^k}{1 - \|\mathbf{B}\|} \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\| \quad (5.4.3)$$

**证明** 由于  $\|\mathbf{B}\| < 1$ , 根据  $\rho(\mathbf{B}) \leq \|\mathbf{B}\|$ , 利用定理 5.4.1, 迭代公式 (5.1.3) 是收敛的, 即

$$\lim_{k \rightarrow \infty} \mathbf{x}^{(k)} = \mathbf{x}^*$$

且

$$\mathbf{x}^* = \mathbf{B}\mathbf{x}^* + \mathbf{f} \quad (5.4.4)$$

由式 (5.1.3) 和式 (5.4.4) 得

$$\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} = \mathbf{B}(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}), \quad \mathbf{x}^{(k+1)} - \mathbf{x}^* = \mathbf{B}(\mathbf{x}^{(k)} - \mathbf{x}^*)$$

利用向量和矩阵范数的性质得

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \leq \|\mathbf{B}\| \cdot \|\mathbf{x}^{(k)} - \mathbf{x}^*\| \quad (5.4.5)$$

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| \leq \|\mathbf{B}\| \cdot \|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\| \quad (5.4.6)$$

反复利用式 (5.4.6), 得

$$\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\| \leq \|\mathbf{B}\|^{k-1} \cdot \|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\| \quad (5.4.7)$$

再利用式 (5.4.5), 有

$$\begin{aligned} \|\mathbf{x}^{(k)} - \mathbf{x}^*\| &= \|\mathbf{x}^{(k)} - \mathbf{x}^{(k+1)} + \mathbf{x}^{(k+1)} - \mathbf{x}^*\| \\ &\leq \|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| + \|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \\ &\leq \|\mathbf{B}\| \cdot \|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\| + \|\mathbf{B}\| \cdot \|\mathbf{x}^{(k)} - \mathbf{x}^*\| \end{aligned}$$

因为  $\|\mathbf{B}\| < 1$ ,  $1 - \|\mathbf{B}\| > 0$ , 所以

$$\|\mathbf{x}^{(k)} - \mathbf{x}^*\| \leq \frac{\|\mathbf{B}\|}{1 - \|\mathbf{B}\|} \|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|$$

式 (5.4.2) 得证, 再利用式 (5.4.7), 可得式 (5.4.3), 定理得证。

在实际计算时, 利用式 (5.4.2), 可将  $\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_{\infty} < \varepsilon$  作为终止计算的条件。

**例 5.4.4** 设  $\mathbf{x} = \mathbf{B}\mathbf{x} + \mathbf{f}$ , 式中

$$\mathbf{B} = \begin{pmatrix} 0.9 & 0 \\ 0.3 & 0.8 \end{pmatrix}, \mathbf{f} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

试说明虽然  $\|\mathbf{B}\| > 1$ , 但迭代公式  $\mathbf{x}^{(k+1)} = \mathbf{B}\mathbf{x}^{(k)} + \mathbf{f}$  是收敛的。

**证明**  $\|\mathbf{B}\|_{\infty} = 1.1$ ,  $\|\mathbf{B}\|_1 = 1.2$ ,  $\|\mathbf{B}\|_F = \sqrt{1.54}$ ,  $\|\mathbf{B}\|_2 = 1.021$ , 故  $\|\mathbf{B}\| > 1$ , 不满足迭代法收敛的充分条件。但  $\det(\lambda \mathbf{I} - \mathbf{B}) = \begin{vmatrix} \lambda - 0.9 & 0 \\ -0.3 & \lambda - 0.8 \end{vmatrix} = (\lambda - 0.9)(\lambda - 0.8) = 0$ , 得  $\lambda_1 = 0.9$ ,  $\lambda_2 = 0.8$ , 故  $\rho(\mathbf{B}) = 0.9 < 1$ , 从而迭代法收敛。

雅可比迭代法的算法框图如图 5.4.1 所示, 图中  $e$  表示  $\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_{\infty}$ ,  $\varepsilon$  为精度控制常数, 当  $e < \varepsilon$  时, 计算终止,  $N$  为最大迭代次数。

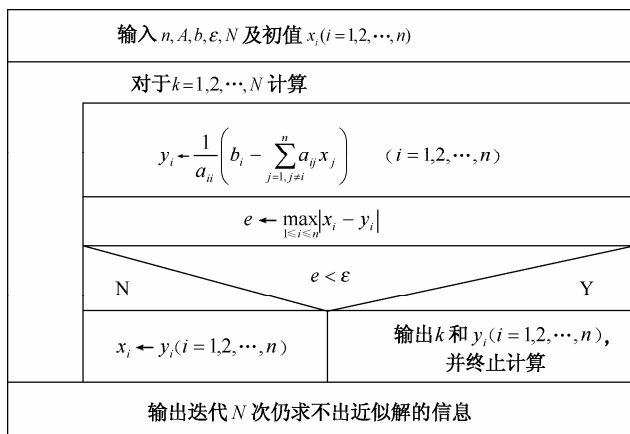


图 5.4.1 雅可比迭代法的算法框图

高斯-塞德尔迭代法的算法框图如图 5.4.2 所示，图中  $t$  用于暂时存放  $x_i$  的旧值，以便计算  $x_i^{(k)} - x_i^{(k-1)}$ 。

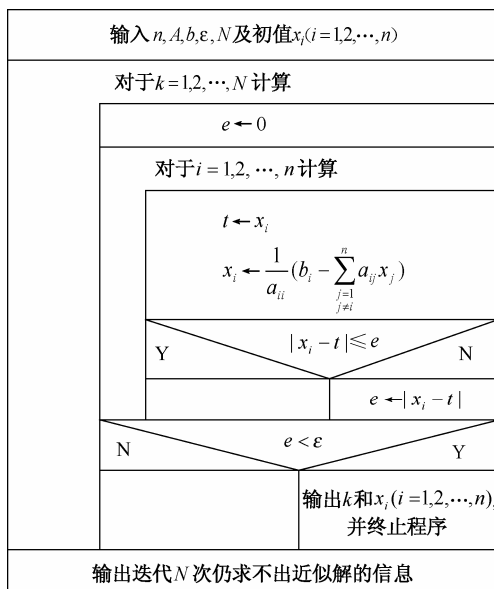


图 5.4.2 高斯-塞德尔迭代法的算法框图

定理 5.4.1 和定理 5.4.3 适用于任何迭代法，当然也适用于雅可比迭代法和高斯-塞德尔迭代法，但对于大型方程组，计算其迭代矩阵和其特征值非常复杂。根据雅可比迭代法和高斯-塞德尔迭代法的特殊性，下面给出一些根据系数矩阵  $A$  的特征来判断这两种迭代法收敛性的充分条件。

**定理 5.4.4 (迭代法收敛的充分条件 2)** 若线性方程组  $Ax = b$  的系数矩阵为严格对角占优或不可约对角占优矩阵，则雅可比迭代法和高斯-塞德尔迭代法收敛。

**证明** 关于对角占优阵的概念，参见 2.3.8 节的定义 2.3.18 至定义 2.3.20，这里只证明  $A$  是严格对角占优阵的情况。

对于雅可比迭代法，由于

$$B_J = D^{-1}(L + D)$$

可以求得

$$\|B_J\|_{\infty} = \max_{1 \leq i \leq n} \sum_{\substack{j=1 \\ j \neq i}}^n \frac{|a_{ij}|}{|a_{ii}|}$$

由  $A$  的严格对角占优定义 2.3.19 得

$$\|B_J\|_{\infty} < 1$$

所以雅可比迭代法收敛。

对于高斯-塞德尔迭代法, 由于

$$B_{G-S} = (D - L)^{-1}U$$

由 2.3.8 节的定理 2.3.4 可知, 若  $A$  为严格对角占优矩阵, 则必有  $a_{ii} \neq 0 (i = 1, 2, \dots, n)$ , 从而

$$\det(D - L)^{-1} = \prod_{i=1}^n \frac{1}{a_{ii}} \neq 0$$

设  $\lambda$  为  $B_{G-S}$  的特征值, 则  $B_{G-S}$  的特征方程为

$$\begin{aligned} \det(\lambda I - B_{G-S}) &= \det(\lambda I - (D - L)^{-1}U) \\ &= \det(D - L)^{-1} \cdot \det(\lambda(D - L) - U) = 0 \end{aligned}$$

从而

$$\det(\lambda(D - L) - U) = 0$$

现在, 用反证法证明  $B_{G-S}$  的特征值  $|\lambda| < 1$ , 假设  $|\lambda| \geq 1$ , 且由于  $A$  是严格对角占优矩阵, 有

$$|\lambda| \cdot |a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^n |\lambda| \cdot |a_{ij}| > \sum_{j=1}^{i-1} |\lambda| \cdot |a_{ij}| + \sum_{j=i+1}^n |a_{ij}| \quad (i = 1, 2, \dots, n)$$

这说明

$$\lambda(D - L) - U = \begin{pmatrix} \lambda a_{11} & a_{12} & \dots & a_{1n} \\ \lambda a_{21} & \lambda a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ \lambda a_{n1} & \lambda a_{n2} & \dots & \lambda a_{nn} \end{pmatrix}$$

是严格对角占优矩阵。

由定理 2.3.4 可知, 它是非奇异矩阵, 即

$$\det(\lambda(D - L) - U) \neq 0$$

这与假设矛盾, 只有  $|\lambda| < 1$ , 才能使

$$\det(\lambda(D - L)) = 0$$

从而

$$\rho(B_{G-S}) < 1$$

这说明高斯-塞德尔迭代法收敛。定理证毕。

在例 5.2.1 和例 5.3.1 中, 由于系数矩阵

$$A = \begin{pmatrix} 10 & -1 & -2 \\ -1 & 10 & -2 \\ -1 & -1 & 5 \end{pmatrix}$$

为严格对角占优阵, 因此解此方程组的雅可比迭代公式和高斯-塞德尔迭代公式收敛。

**定理 5.4.5** 若线性方程组  $Ax = b$  的系数矩阵  $A$  对称正定, 则高斯-塞德尔迭代法收敛; 若

$A$  对称正定,  $2D-A$  也对称正定 ( $D$  为  $A$  的对角元组成的对角阵,  $2D-A$  与  $A$  只是非对角元的符号不同), 则雅可比迭代法收敛; 若  $A$  对称正定, 而  $2D-A$  非正定, 则雅可比迭代法不收敛。(证明略。)

例 5.4.5 设在线性方程组  $Ax=b$  中

$$A = \begin{pmatrix} 1 & 0.9 & 0.9 \\ 0.9 & 1 & 0.9 \\ 0.9 & 0.9 & 1 \end{pmatrix}$$

讨论雅可比迭代法和高斯-塞德尔迭代法的收敛性。

解 因为  $A$  对称, 且各阶顺序主子式都大于零, 所以  $A$  对称正定, 由定理 5.2.4 可知, 用高斯-塞德尔迭代法解此方程组收敛, 但在

$$2D-A = \begin{pmatrix} 1 & -0.9 & -0.9 \\ -0.9 & 1 & -0.9 \\ -0.9 & -0.9 & 1 \end{pmatrix}$$

中,  $\det(2D-A) < 0$ , 所以  $2D-A$  非正定, 由定理 5.4.4 可知, 用雅可比迭代法不收敛。

## 本章小结

本章介绍的解线性方程组的迭代法主要用于解高阶稀疏矩阵方程组, 其特点是: 占用内存少、程序设计简单、原始系数矩阵在计算过程中始终不变。但是, 存在收敛性和收敛速度的问题。

雅可比迭代法也称简单迭代法其基本思想是从方程组的第  $i$  个方程求出  $x_i$ , 并建立相应的迭代公式求  $x_i^{(k+1)}$ 。高斯-塞德尔迭代法在雅可比迭代法的基础上进行了改进, 在求  $x_i^{(k+1)}$  时, 用已求出的  $x_1 \sim x_{i-1}$  的新值代替旧值, 因此也称异步迭代法。在二者都收敛时, 高斯-塞德尔迭代公式收敛较快, 所以, 应用也较广泛。

在迭代法的收敛性定理中, 迭代法收敛性基本定理是充分必要条件, 其余都是充分条件。

## 习题 5

$$5.1 \quad \text{对方程组} \begin{cases} x_1 + 2x_2 - 2x_3 = 1 \\ x_1 + x_2 + x_3 = 3 \\ 2x_1 + 2x_2 + x_3 = 5 \end{cases} \text{ 用雅可比迭代法求解是否收敛? 若收敛, 取初始向量}$$

$x^{(0)} = (0, 0, 0)^T$ , 迭代计算至  $\|x^{(k+1)} - x^{(k)}\|_{\infty} \leq 10^{-8}$ 。

5.2 设线性方程组  $Ax=b$  的系数矩阵为

$$A = \begin{pmatrix} a & 1 & 3 \\ 1 & a & 2 \\ -3 & 2 & a \end{pmatrix}$$

试求能使雅可比迭代法收敛的  $a$  的取值范围。

5.3 实数  $a \neq 0$ , 考察矩阵  $A = \begin{pmatrix} 1 & a & 0 \\ a & 1 & a \\ 0 & a & 1 \end{pmatrix}$ , 试就方程组  $Ax = b$  建立雅可比迭代法和高斯-

塞德尔迭代法的计算公式, 讨论  $a$  取何值时迭代收敛。

5.4 解方程组  $\begin{cases} a_{11}x_1 + a_{12}x_2 = b_1 \\ a_{21}x_1 + a_{22}x_2 = b_2 \end{cases}$  的雅可比迭代公式为

$$\begin{cases} x_1^{(k)} = \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(k-1)}) \\ x_2^{(k)} = \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(k-1)}) \end{cases} \quad (k=1, 2, \dots)$$

求证: 上述公式产生的向量序列  $|x_1^{(k)}|$  收敛的充要条件是  $r = \left| \frac{a_{12}a_{21}}{a_{11}a_{22}} \right| < 1$ 。

5.5 对于线性方程组  $Ax = b$ , 设系数矩阵为  $A = \begin{pmatrix} \lambda & -2 & 2 \\ -1 & \lambda & -1 \\ -2 & -2 & \lambda \end{pmatrix}$ , 试问:  $\lambda$  取什么值时, 雅

可比迭代法收敛。

5.6 设  $A = (a_{ij}) \in R^{n \times n}$ , 有“列严格对角占优性质”, 即  $|a_{jj}| > \sum_{i=1, i \neq j}^n |a_{ij}|$  ( $j=1, 2, \dots, n$ ), 证明:

解方程组  $Ax = b$  的雅可比迭代公式收敛。

5.7 设  $A = \begin{pmatrix} 1 & a & a \\ a & 1 & a \\ a & a & 1 \end{pmatrix}$ , 求解方程组  $Ax = b$ , 证明: 当  $-\frac{1}{2} < a < 1$  时高斯-塞德尔迭代法收敛,

而雅可比迭代法只对  $-\frac{1}{2} < a < \frac{1}{2}$  才收敛。

5.8 (1) 设迭代公式  $x^{(k+1)} = Bx^{(k)} + f$  ( $k=0, 1, \dots$ ), 若迭代矩阵  $B \in R^{n \times n}$  的谱半径  $\rho(B) = 0$ , 则对任意  $x^{(0)} \in R^n$ ,  $x^{(n)}$  一定是  $x = Bx + f$  的解向量, 即  $x^{(n)} = x$ 。

(2) 设  $x^{(k+1)} = Bx^{(k)} + f$  ( $k=0, 1, \dots$ ), 式中  $B = \begin{pmatrix} 0 & -2 & 2 \\ -1 & 0 & -1 \\ -2 & -2 & 0 \end{pmatrix}$ ,  $f = \begin{pmatrix} 1 \\ 3 \\ 5 \end{pmatrix}$ , 求  $\rho(B)$  并计算

$x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}$ 。

5.9 设方程组  $\begin{cases} 5x_1 + 2x_2 + x_3 = -12 \\ -x_1 + 4x_2 + 2x_3 = 20 \\ 2x_1 - 3x_2 + 10x_3 = 3 \end{cases}$

(1) 考察用雅可比迭代法和高斯-塞德尔迭代法解此方程组的收敛性。

(2) 写出用雅可比迭代法及高斯-塞德尔迭代法解此方程组的迭代公式并从  $x^{(0)} = (0, 0, 0)^T$  计算到  $\|x^{(k+1)} - x^{(k)}\|_{\infty} < 10^{-4}$  为止。

5.10 设方程组  $\begin{cases} a_{11}x_1 + a_{12}x_2 = b_1 \\ a_{21}x_1 + a_{22}x_2 = b_2 \end{cases}$  ( $a_{11}, a_{22} \neq 0$ ), 证明解此方程的雅可比迭代公式和高斯-塞德

尔迭代公式同时收敛或发散。

5.11 下列两个方程组  $\mathbf{Ax} = \mathbf{b}$ , 若分别用雅可比迭代法及高斯-塞德尔迭代法求解, 是否收敛?

$$(1) \mathbf{A} = \begin{pmatrix} 1 & 2 & -2 \\ 1 & 1 & 1 \\ 2 & 2 & 1 \end{pmatrix} \quad (2) \mathbf{A} = \begin{pmatrix} 2 & -1 & 1 \\ 2 & 2 & 2 \\ -1 & -1 & 2 \end{pmatrix}$$

5.12 设  $\mathbf{A} = \begin{pmatrix} 10 & a & 0 \\ b & 10 & b \\ 0 & a & 5 \end{pmatrix}$ ,  $\det(\mathbf{A}) \neq 0$ , 用  $a, b$  表示解方程组  $\mathbf{Ax} = \mathbf{f}$  的雅可比迭代公式及高

斯-塞德尔迭代公式收敛的充分必要条件。

5.13 用迭代公式  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + a(\mathbf{Ax}^{(k)} - \mathbf{b})$  求解方程组  $\mathbf{Ax} = \mathbf{b}$ , 问取什么实数  $a$  可使迭代收敛? 证明:  $a = -0.4$  时收敛最快。已知:  $\mathbf{A} = \begin{pmatrix} 3 & 2 \\ 1 & 2 \end{pmatrix}$ ,  $\mathbf{b} = \begin{pmatrix} 3 \\ -1 \end{pmatrix}$ 。

5.14 设方程组  $\mathbf{Ax} = \mathbf{b}$  ( $a_{ii} \neq 0, i=1, 2, \dots, n$ ), 证明:

(1) 解此方程组的雅可比迭代公式收敛的充分必要条件是  $\det \begin{pmatrix} a_{11}\lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22}\lambda & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn}\lambda \end{pmatrix} = 0$  的

根模  $|\lambda| < 1$ 。

(2) 解此方程组的高斯-塞德尔迭代公式收敛的充分必要条件是  $\det \begin{pmatrix} a_{11}\lambda & a_{12} & \cdots & a_{1n} \\ a_{21}\lambda & a_{22}\lambda & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}\lambda & a_{n2}\lambda & \cdots & a_{nn}\lambda \end{pmatrix} = 0$

的根模  $|\lambda| < 1$ 。

5.15 已知方程组  $\begin{pmatrix} 1 & 2 \\ 0.32 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$ , 解此方程组的雅可比迭代公式是否收敛? 它的收敛速度  $R(\mathbf{B})$  是多少?

5.16 设方程组  $\mathbf{Ax} = \mathbf{b}$ , 式中,  $\mathbf{A} = \begin{pmatrix} 2 & -1 \\ 1 & 1.5 \end{pmatrix}$ , 写出雅可比迭代法的迭代矩阵和高斯-塞德尔迭代法的迭代矩阵。

5.17 用高斯-塞德尔迭代法求解方程组  $\begin{cases} x_1 + ax_2 = 4 \\ 2ax + x_2 = -3 \end{cases}$ , 其中  $a$  为实数, 其收敛的充要条件是  $a$  满足什么条件?

## 第6章 函数插值



### 学习要点

- (1) 插值的基本概念, 包括线性插值、抛物插值 and 多项式插值的存在唯一性。
- (2) 多项式插值方法, 包括基于基函数的拉格朗日插值、基于差商的牛顿插值和基于导数的埃尔米特插值。
- (3) 分段插值, 包括分段线性插值、分段埃尔米特插值和样条插值。



### 教学建议

本章的重点是多项式插值, 要求学生了解插值的概念和插值多项式的存在唯一性, 熟练掌握拉格朗日插值和牛顿插值方法, 并利用余项定理进行误差估计, 了解两点三次埃尔米特插值在推导三次样条插值多项式中的作用。本章内容较多, 样条函数插值的推导过程可作为选学内容。建议学时为 8~10 学时。

## 6.1 引言

### 6.1.1 插值问题

函数常被用来描述客观事物变化的内在规律(数量关系), 如天体运动, 气候变化, 股市波动等。但在生产和科研实践中遇到的大量函数, 却是复杂函数, 其表现形式有三类。第一类是函数表的形式, 这些函数是通过实验或观测得到的一些数据, 这些数据只是某些离散点  $x_i$  上的值(包括函数值  $f(x_i)$ , 导数值  $f'(x_i)$  等,  $i = 0, 1, 2, \dots, n$ ), 虽然其函数关系是客观存在的, 但却不知道具体的解析表达式, 因此不便于分析研究这类数表函数的性质, 也不能直接得出其他未列出点的函数值。第二类是图形、图像的形式, 如飞机和船舶的外形, 计算机图片和动画等。为了研究飞机在空气中的阻力, 船舶在水中的阻力, 就要对其外形进行很好的分析。为了提高图片的质量和动画效果, 就要对原有图片和动画进行修改。由于实际中的图形、图像没有具体的函数或解析式, 因此对其进行研究非常困难。第三类是有解析式的复杂函数, 虽然它们有解析式, 但因其过于复杂, 不便于计算和分析。

对于实际中的这些复杂函数, 我们希望构造一个既能反映函数本身的特性, 又便于计算的简单函数, 近似代替原来的函数。其主要思想是对于一组离散点  $(x_i, f(x_i))$ ,  $i = 0, 1, 2, \dots, n$ , 选定一个便于计算的函数形式  $p(x)$ , 如多项式函数、分段线性函数、有理函数、三角函数等。要求简单函数  $p(x)$  满足  $p(x_i) = f(x_i)$ ,  $i = 0, 1, 2, \dots, n$ 。由此确定函数  $p(x)$  作为  $f(x)$  的近似函数, 这就是函数插值方法。

用插值方法求函数的近似表达式时, 首先要选定函数的形式。可供选择的函数很多, 最常



用的是多项式函数, 因为多项式函数计算简便, 只需用加、减、乘运算, 便于编程计算, 而且其导数与积分仍为多项式函数。用多项式函数作为研究插值的工具, 称为代数插值。因此代数插值问题可以定义如下。

**定义 6.1.1** 设  $y = f(x)$  是在区间  $[a, b]$  内的连续函数, 记为:  $f \in C[a, b]$ , 已知  $f(x)$  在  $[a, b]$  内  $n+1$  个互异点  $a \leq x_0 < x_1 < \cdots < x_n \leq b$  处的函数值为:  $y_i = f(x_i)$ ,  $i = 0, 1, 2, \cdots, n$ , 若有不超过  $n$  次的多项式  $p_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$ , 满足条件

$$p_n(x_i) = y_i \quad (i = 0, 1, 2, \cdots, n) \quad (6.1.1)$$

则称  $p_n(x)$  为函数  $f(x)$  在区间  $[a, b]$  内通过点列  $\{x_i, y_i\}_{i=0}^n$  的插值多项式, 如图 6.1.1 所示。其中,  $[a, b]$  称为插值区间,  $\{x_i\}_{i=0}^n$  称为插值节点, 求函数值  $f(x)$  的点  $x (x \neq x_i)$  称为插值点。  $f(x)$  称为被插函数,  $p_n(x)$  称为插值函数, 式 (6.1.1) 称为插值条件,  $f(x) - p_n(x)$  称为插值余项 (也称误差)。

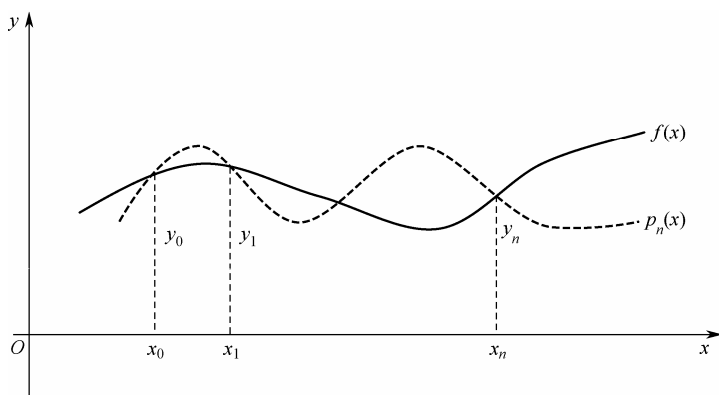


图 6.1.1 插值函数与被插函数

函数插值是数值计算的基本问题, 所涉及的问题有: 存在唯一性、构造方法、截断误差和收敛性, 以及数值计算的稳定性。

## 6.1.2 插值多项式的存在唯一性

在代数插值中, 首先要解决的问题是: 满足插值条件式 (6.1.1) 的插值多项式  $p_n(x)$  是否存在? 如果存在, 是否唯一?  $n$  次多项式  $p_n(x)$  有  $n+1$  个待定系数  $a_0, a_1, \cdots, a_n$ , 利用给出的  $n+1$  个不同的节点  $x_0, x_1, \cdots, x_n$ , 由插值条件式 (6.1.1), 可得关于系数  $a_0, a_1, \cdots, a_n$  的  $n+1$  个方程

$$\begin{cases} a_0 + a_1x_0 + a_2x_0^2 + \cdots + a_nx_0^n = y_0 \\ a_0 + a_1x_1 + a_2x_1^2 + \cdots + a_nx_1^n = y_1 \\ \vdots \\ a_0 + a_1x_n + a_2x_n^2 + \cdots + a_nx_n^n = y_n \end{cases} \quad (6.1.2)$$

由于方程组 (6.1.2) 的系数行列式为范德蒙 (Vandermode) 行列式, 即

$$\det(A) = \begin{vmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{vmatrix} = \prod_{i=1}^n \prod_{j=0}^{i-1} (x_i - x_j) \quad (6.1.3)$$

当  $x_0, x_1, \dots, x_n$  互不相同时, 行列式 (6.1.3) 的值不为 0 (参见例 2.3.3), 从而方程组 (6.1.2) 存在唯一的解  $a_0, a_1, \dots, a_n$ , 于是有下面的定理。

**定理 6.1.1 (存在唯一性定理)** 当插值节点互异时, 满足插值条件式 (6.1.1) 的  $n$  次插值多项式  $p_n(x)$  存在且唯一。

由定理 6.1.1 可以看出:

① 不论用什么方法去构造, 也不论用什么形式来表示插值多项式  $p_n(x)$ , 只要满足插值条件式 (6.1.1), 其结果都是互相恒等的, 当然, 其余项 (误差) 也是相同的。

② 要构造插值多项式  $p_n(x)$  可以通过求方程组 (6.1.2) 的解  $a_0, a_1, \dots, a_n$  得到。但这样做, 当  $n$  较大时, 不但计算工作量大, 而且难以得到  $p_n(x)$  的简单表达式。因此, 在实际中不采用这种方法, 而采用更简单的方法。

## 6.2 拉格朗日插值

### 6.2.1 线性插值与抛物插值

#### 1. 线性插值

最简单的插值问题是已知两点  $(x_0, y_0)$  及  $(x_1, y_1)$ , 这里  $f(x_0) = y_0$ ,  $f(x_1) = y_1$ 。通过两点  $(x_0, y_0)$  及  $(x_1, y_1)$  的插值多项式是一条直线, 两点式的直线方程为

$$L_1(x) = \frac{x - x_1}{x_0 - x_1} y_0 + \frac{x - x_0}{x_1 - x_0} y_1$$

显然,  $L_1(x_0) = y_0$ ,  $L_1(x_1) = y_1$  满足插值条件, 所以  $L_1(x)$  就是线性插值函数。

若令

$$l_0(x) = \frac{x - x_1}{x_0 - x_1}, \quad l_1(x) = \frac{x - x_0}{x_1 - x_0}$$

则有

$$L_1(x) = y_0 l_0(x) + y_1 l_1(x) \quad (6.2.1)$$

这里  $l_0(x)$  和  $l_1(x)$  分别看做满足条件

$$\begin{cases} l_0(x_0) = 1, & l_0(x_1) = 0 \\ l_1(x_0) = 0, & l_1(x_1) = 1 \end{cases} \quad (6.2.2)$$

的插值多项式, 称  $l_0(x)$ ,  $l_1(x)$  为关于  $x_0, x_1$  的线性插值基函数, 如图 6.2.1 所示。

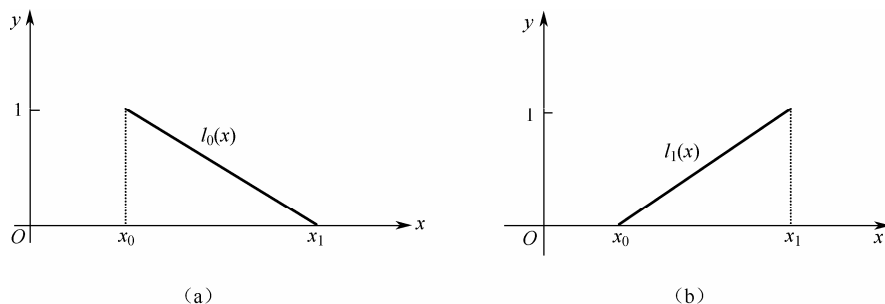


图 6.2.1 线性插值基函数  $l_0(x)$ ,  $l_1(x)$

例 6.2.1 已知:  $\sqrt{100}=10$ ,  $\sqrt{121}=11$ , 求  $y=\sqrt{115}$ 。

解 设  $x_0=100, y_0=10, x_1=121, y_1=11$ , 令  $x=115$ , 代入式 (6.2.1), 求得

$$y=\sqrt{115} \approx \frac{115-121}{100-121} \times 10 + \frac{115-100}{121-100} \times 11 = 10.71428$$

可以验证这个结果有 3 位有效数字。

式 (6.2.1) 表明, 线性插值函数  $L_1(x)$  可以通过插值基函数  $l_0(x)$  和  $l_1(x)$  的线性组合得到, 且组合系数恰为所给数据  $y_0, y_1$ 。

## 2. 抛物插值

线性插值仅仅利用两个节点的信息, 精度肯定很低, 为了提高插值函数的精度, 再增加一个点, 即已知三个点:

$$(x_0, y_0), (x_1, y_1), (x_2, y_2) \quad (\text{这里 } f(x_i) = y_i, i = 0, 1, 2)$$

构造通过这三个点的二次插值多项式  $L_2(x)$ , 使

$$L_2(x_i) = y_i \quad (i = 0, 1, 2)$$

$L_2(x)$  的几何意义是过这三点的抛物线, 因此, 插值函数  $L_2(x)$  也称为抛物插值函数。

为了求出插值多项式  $L_2(x)$ , 可假设

$$L_2(x) = y_0 l_0(x) + y_1 l_1(x) + y_2 l_2(x) \quad (6.2.3)$$

式中  $l_0(x), l_1(x), l_2(x)$  为待定函数。由插值条件

$$L_2(x_0) = y_0, \quad L_2(x_1) = y_1, \quad L_2(x_2) = y_2$$

可知, 待定函数  $l_0(x), l_1(x), l_2(x)$  满足如下性质

$$\begin{cases} l_0(x_0) = 1, l_0(x_1) = 0, l_0(x_2) = 0 \\ l_1(x_0) = 0, l_1(x_1) = 1, l_1(x_2) = 0 \\ l_2(x_0) = 0, l_2(x_1) = 0, l_2(x_2) = 1 \end{cases} \quad (6.2.4)$$

下面以  $l_0(x)$  为例, 求待定函数。

由式 (6.2.4) 可知,  $x_1, x_2$  是  $l_0(x)$  的两个零点, 因而设

$$l_0(x) = c(x - x_1)(x - x_2)$$

再由条件  $l_0(x_0) = 1$  确定系数

$$c = \frac{1}{(x_0 - x_1)(x_0 - x_2)}$$

结果得

$$l_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}$$

同理可得

$$l_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}$$

$$l_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

于是, 二次插值函数  $L_2(x)$  为

$$L_2(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} y_0 + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} y_1 + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} y_2 \quad (6.2.5)$$

由式 (6.2.5) 可知,  $L_2(x_0)=y_0$ ,  $L_2(x_1)=y_1$ ,  $L_2(x_2)=y_2$ , 满足插值条件, 所以式 (6.2.5) 就是满足插值条件的二次插值函数。

我们称满足式 (6.2.4) 的函数  $l_0(x), l_1(x), l_2(x)$  为关于  $x_0, x_1, x_2$  的二次插值基函数, 如图 6.2.2 所示。

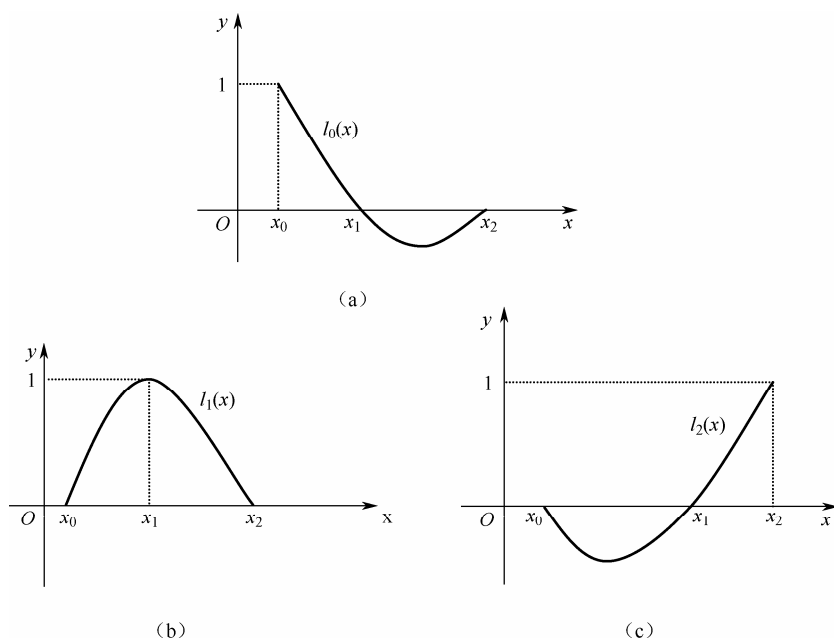


图 6.2.2 二次插值基函数  $l_0(x), l_1(x), l_2(x)$

**例 6.2.2** 已知:  $\sqrt{100}=10$ ,  $\sqrt{121}=11$ ,  $\sqrt{144}=12$ , 求  $y=\sqrt{115}$ 。

**解** 设  $x_0=100, y_0=10, x_1=121, y_1=11, x_2=144, y_2=12$ , 令  $x=115$ , 代入式 (6.2.5), 求得

$$y=\sqrt{115} \approx \frac{(115-121)(115-144)}{(100-121)(100-144)} \times 10 + \frac{(115-100)(115-144)}{(121-100)(121-144)} \times 11 + \frac{(115-100)(115-121)}{(144-100)(144-121)} \times 12 = 10.7228$$

可以验证, 这个结果具有 4 位有效数字。

## 6.2.2 拉格朗日插值

将线性插值和抛物插值推广到一般情形, 现在考虑过  $n+1$  个点  $(x_i, y_i)$ ,  $i=0, 1, 2, \dots, n$ , 的插值多项式  $L_n(x)$  的问题。

已知  $n+1$  个点  $(x_i, y_i)$ ,  $i=0, 1, 2, \dots, n$ , 构造  $n$  次插值多项式  $L_n(x)$ , 使

$$L_n(x_i) = y_i \quad (i=0, 1, 2, \dots, n)$$

用插值基函数的方法, 可设

$$L_n(x) = \sum_{i=0}^n y_i l_i(x) \quad (6.2.6)$$

式中,  $l_i(x)$  称为关于  $x_0, x_1, \dots, x_n$  的插值基函数, 它满足条件

$$l_i(x_j) = \begin{cases} 1 & j=i \\ 0 & j \neq i \end{cases} \quad (i, j=0, 1, 2, \dots, n) \quad (6.2.7)$$

这表明, 除  $x_i$  外的所有节点都是  $l_i(x)$  的零点, 故

$$l_i(x) = c \prod_{\substack{j=0 \\ j \neq i}}^n (x - x_j)$$

再由条件  $l_i(x_i) = 1$  确定其系数  $c$ ，结果为

$$l_i(x) = \frac{(x - x_0) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)} = \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j} \quad (6.2.8)$$

这里， $\prod$  的含义是累乘， $\prod_{\substack{j=0 \\ j \neq i}}^n$  表示乘积遍取下标  $j$  从 0 到  $n$  且除  $i$  以外的全部值，于是  $n$  次插值

多项式  $L_n(x)$  为

$$L_n(x) = \sum_{i=0}^n \left( \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j} \right) y_i \quad (6.2.9)$$

因为每个插值基函数  $l_i(x)$  都是  $n$  次的， $L_n(x)$  的次数不会超过  $n$ ，且由式 (6.2.7) 可得

$$L_n(x_k) = \sum_{i=0}^n l_i(x_k) y_i = y_k$$

所以  $L_n(x)$  满足插值条件。因此，式 (6.2.9) 即为  $n$  次插值多项式，称为拉格朗日 (Lagrange) 插值多项式。该公式形式对称，结构紧凑，所以，非常便于编写程序。式 (6.2.9) 在逻辑结构上表现为二重循环，内循环 ( $j$ ) 累乘求得  $l_i(x)$ ，然后通过外循环 ( $i$ ) 累加得出插值结果  $y$ 。拉格朗日插值方法的算法框图如图 6.2.3 所示。

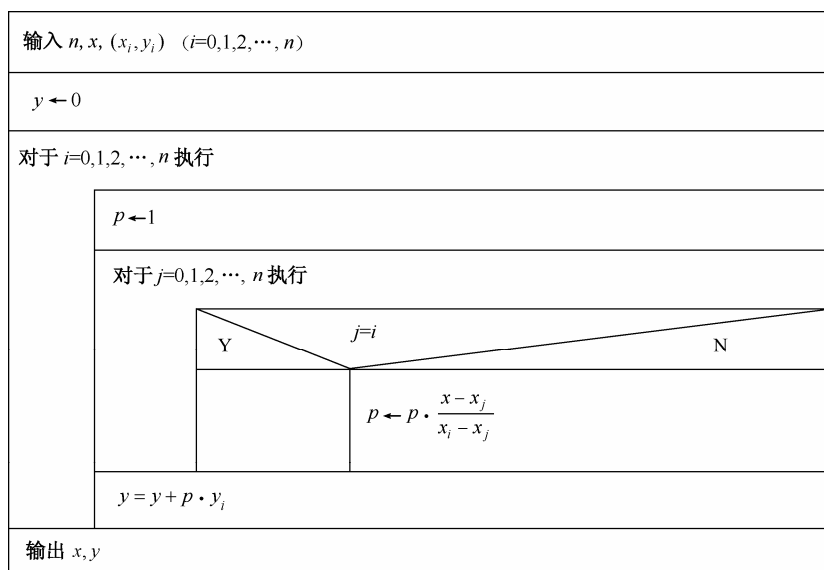


图 6.2.3 拉格朗日插值方法的算法框图

**例 6.2.3** 已知  $x = 1, 2, 3, 4, 5$  对应的函数值为  $f(x) = 1, 4, 7, 8, 6$ ，试构造 4 次拉格朗日插值多项式，并求  $f(1.5)$  的近似值。

**解** 由式 (6.2.9) 直接可得

$$L_4(x) = \frac{(x-2)(x-3)(x-4)(x-5)}{(1-2)(1-3)(1-4)(1-5)} \times 1 + \frac{(x-1)(x-3)(x-4)(x-5)}{(2-1)(2-3)(2-4)(2-5)} \times 4 +$$

$$\frac{(x-1)(x-2)(x-4)(x-5)}{(3-1)(3-2)(3-4)(3-5)} \times 7 + \frac{(x-1)(x-2)(x-3)(x-5)}{(4-1)(4-2)(4-3)(4-5)} \times 8 +$$

$$\frac{(x-1)(x-2)(x-3)(x-4)}{(5-1)(5-2)(5-3)(5-4)} \times 6 = \dots = \frac{1}{24}x^4 - \frac{3}{4}x^3 + \frac{83}{24}x^2 - \frac{11}{4}x + 1$$

所以

$$f(1.5) \approx L_4(1.5) = \frac{299}{128} = 2.3359375$$

### 6.2.3 插值余项与误差估计

若  $f(x)$  在  $[a, b]$  内的插值多项式为  $L_n(x)$ , 则称  $R_n(x) = f(x) - L_n(x)$  为  $L_n(x)$  的插值余项 (也称误差)。

**定理 6.2.1 (余项定理)** 设  $f(x)$  在  $[a, b]$  内的  $n+1$  阶导数连续, 记为  $f(x) \in C^{n+1}[a, b]$ , 并且  $f(x)$  在互异节点  $a \leq x_0 < x_1 < \dots < x_n \leq b$  的函数值为  $y_0, y_1, \dots, y_n$ 。若满足插值条件  $L_n(x_i) = y_i$  ( $i = 0, 1, 2, \dots, n$ ) 的插值多项式为  $L_n(x)$ , 则对  $\forall x \in [a, b]$  有

$$R_n(x) = f(x) - L_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \cdot \prod_{j=0}^n (x - x_j) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \cdot \omega_{n+1}(x) \quad (6.2.10)$$

式中  $a < \xi < b$ ,  $\omega_{n+1}(x) = \prod_{j=0}^n (x - x_j)$

**证明** 由插值条件知  $R_n(x_i) = 0$  ( $i = 0, 1, \dots, n$ ), 故对  $\forall x \in [a, b]$  有

$$R_n(x) = k(x)(x - x_0)(x - x_1) \cdots (x - x_n) = k(x)\omega_{n+1}(x) \quad (6.2.11)$$

式中,  $k(x)$  是依赖于  $x$  的待定函数, 将  $x \in [a, b]$  看做在  $[a, b]$  内的任一固定点, 可设辅助函数

$$\varphi(t) = f(t) - L_n(t) - k(x)(t - x_0)(t - x_1) \cdots (t - x_n)$$

则

$$\varphi(x_i) = 0 \quad (i = 0, 1, 2, \dots, n) \quad \text{且} \quad \varphi(x) = 0$$

这说明  $\varphi(t)$  在  $[a, b]$  内有  $n+2$  个零点  $x_0, x_1, \dots, x_n$  及  $x$ , 由罗尔 (Rolle) 定理可知,  $\varphi'(t)$  在  $[a, b]$  内至少有  $n+1$  个零点。反复应用罗尔定理, 可得  $\varphi^{(n+1)}(t)$  在  $[a, b]$  内至少有一个零点  $\xi \in (a, b)$ , 使

$$\varphi^{(n+1)}(\xi) = f^{(n+1)}(\xi) - (n+1)!k(x) = 0$$

于是

$$k(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}$$

代入式 (6.2.11), 得余项表达式 (6.2.10), 证毕。

由于定理 6.2.1 中的  $\xi$  无法确定, 因此式 (6.2.10) 的准确值难以计算。不过, 因为  $f^{(n+1)}(x)$  在闭区间  $[a, b]$  内连续, 由于闭区间内的连续函数必有界, 所以必有最大值  $M_{n+1}$ , 即

$$\left| f^{(n+1)}(\xi) \right| \leq \left| \max_{a \leq x \leq b} f^{(n+1)}(x) \right| \leq M_{n+1}$$

于是, 可得误差估计式

$$\left| R_n(x) \right| \leq \frac{M_{n+1}}{(n+1)!} \left| \prod_{j=0}^n (x - x_j) \right| \quad (6.2.12)$$

当  $n=1$  时, 线性插值式 (6.2.1) 的误差估计式为

$$|R_1(x)| \leq \frac{M_2}{2!} |(x-x_0)(x-x_1)| \leq \frac{M_2}{8} (x_1-x_0)^2 \quad (6.2.13)$$

当  $n=2$  时, 抛物插值式 (6.2.5) 的误差估计式为

$$|R_2(x)| \leq \frac{M_3}{3!} |(x-x_0)(x-x_1)(x-x_2)| \quad (6.2.14)$$

**例 6.2.4** 设  $f(x) \in C^2[a, b]$ , 已知插值节点  $x_0 = a, x_1 = b$ , 证明:  $f(x)$  在  $[a, b]$  内的线性插值函数  $L_1(x)$  的误差界为

$$\max_{a \leq x \leq b} |f(x) - L_1(x)| \leq \frac{(b-a)^2}{8} \max_{a \leq x \leq b} |f''(x)|$$

并举例说明上述不等式的等号成立。

**证明** 由插值余项公式, 可得  $f(x) - L_1(x) = \frac{f''(\xi)}{2!} (x-a)(x-b)$ ,  $a < \xi < b$ 。

由于当  $x \in [a, b]$  时, 函数  $|(x-a)(x-b)|$  在  $x = \frac{a+b}{2}$  处取得最大值为  $\frac{(b-a)^2}{4}$ , 因此

$$\begin{aligned} \max_{a \leq x \leq b} |f(x) - L_1(x)| &\leq \frac{1}{2} \max_{a \leq x \leq b} |(x-a)(x-b)| \max_{a \leq x \leq b} |f''(x)| \\ &= \frac{(b-a)^2}{8} \max_{a \leq x \leq b} |f''(x)| \end{aligned}$$

取  $f(x) = x$ , 则有  $L_1(x) = x, f''(x) = 0$ , 因此

$$\max_{a \leq x \leq b} |f(x) - L_1(x)| = 0, \quad \frac{(b-a)^2}{8} \max_{a \leq x \leq b} |f''(x)| = 0$$

此时, 不等式的等号成立。

**例 6.2.5** 已知  $\sqrt{100} = 10$ ,  $\sqrt{121} = 11$ ,  $\sqrt{144} = 12$ , 试对例 6.2.1 中的线性插值和例 6.2.2 中的抛物插值进行误差估计。

**解** 由题意知被插函数为  $y = f(x) = \sqrt{x}$ , 所以

$$f'(x) = \frac{1}{2} x^{-\frac{1}{2}}, \quad f''(x) = -\frac{1}{4} x^{-\frac{3}{2}}, \quad f'''(x) = \frac{3}{8} x^{-\frac{5}{2}} \quad (x \in [100, 144])$$

$$M_2 = \max_{100 \leq x \leq 121} \left| -\frac{1}{4} x^{-\frac{3}{2}} \right| \leq \frac{1}{4000}, \quad M_3 = \max_{100 \leq x \leq 144} \left| \frac{3}{8} x^{-\frac{5}{2}} \right| \leq \frac{3}{8} \times \frac{1}{100000}$$

于是

$$|R_1(x)| \leq \left| \frac{1}{2} \times \frac{1}{4000} \times (115-100) \times (115-121) \right| = 0.01125 < 0.05$$

这说明例 6.2.1 中  $\sqrt{115}$  的近似值 10.71428 具有 3 位有效数字。

$$\begin{aligned} |R_2(x)| &\leq \frac{1}{6} \times \frac{3}{800000} \times |(115-100) \times (115-121) \times (115-144)| \\ &= \frac{45 \times 29}{800000} = 0.00163125 \leq 0.005 \end{aligned}$$

这说明例 6.2.2 中  $\sqrt{115}$  的近似值 10.7228 具有 4 位有效数字。

**例 6.2.6** 设  $f(x) = x^4$ , 试利用拉格朗日插值余项定理写出以 -1, 0, 1, 2 为插值节点的三次插值多项式。

**解** 设由已知插值节点确定的三次插值多项式为  $L_3(x)$ , 根据拉格朗日插值余项定理, 有

$$f(x) - L_3(x) = \frac{f^{(4)}(\xi)}{4!}(x-x_0)(x-x_1)(x-x_2)(x-x_3) \quad (\xi \in (-1, 2))$$

由于  $f(x) = x^4$ , 则  $f^{(4)}(x) = 4!$ , 故  $f(x) - L_3(x) = (x+1)(x-0)(x-1)(x-2)$

即

$$L_3(x) = x^4 - x(x+1)(x-1)(x-2) = 2x^3 + x^2 - 2x$$

**例 6.2.7** 设  $x_j$  为互异节点 ( $j = 0, 1, \dots, n$ ),  $l_j(x)$  为  $n$  次插值基函数, 证明:

$$(1) \sum_{j=0}^n l_j(x) = 1$$

$$(2) \sum_{j=0}^n l_j(0)x_j^k = \begin{cases} 1 & k=0 \\ 0 & k=1, 2, \dots, n \\ (-1)^n x_0 x_1 \cdots x_n & k=n+1 \end{cases}$$

**证明** (1) 对拉格朗日插值多项式, 令  $f(x) = 1$ , 则得

$$1 = \sum_{j=0}^n l_j(x) \times 1 + R_n(x)$$

因为  $R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega_{n+1}(x) = 0$ , 故有  $\sum_{j=0}^n l_j(x) = 1$ 。

(2) 仍由  $f(x) = \sum_{j=0}^n l_j(x)f(x_j) + \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega_{n+1}(x)$ , 令  $f(x) = x^k$ , 则

当  $k=0, f(x)=1$  时, 有  $\sum_{j=0}^n l_j(0) = 1$

当  $k=1, 2, \dots, n$  时,  $f^{(n+1)}(\xi) = 0$ , 故有  $\sum_{j=0}^n l_j(x)x_j^k = x^k$

当  $x=0$  时, 得  $\sum_{j=0}^n l_j(0)x_j^k = 0$

当  $k=n+1$  时,  $f^{(n+1)}(\xi) = (n+1)!$ ,  $x^{n+1} = \sum_{j=0}^n l_j(x)x_j^{n+1} + \omega_{n+1}(x)$

当  $x=0$  时, 得  $\sum_{j=0}^n l_j(0)x_j^k = -(-x_0)(-x_1)\cdots(-x_n) = (-1)^n x_0 x_1 \cdots x_n$

关于插值余项定理 6.2.1 的几点说明如下。

① 若令  $\omega_{n+1}(x) = (x-x_0)(x-x_1)\cdots(x-x_n)$ , 则

$$\omega'_{n+1}(x_i) = (x_i - x_0)(x_i - x_1)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n)$$

式 (6.2.8) 的基函数  $l_i(x)$  可写为

$$l_i(x) = \frac{\omega_{n+1}(x)}{(x-x_i)\omega'_{n+1}(x_i)} \quad (6.2.15)$$

式 (6.2.9) 的拉格朗日插值多项式可写为

$$L_n(x) = \sum_{i=0}^n \frac{\omega_{n+1}(x)}{(x-x_i)\omega'_{n+1}(x_i)} y_i \quad (6.2.16)$$

式 (6.2.10) 的余项公式  $R_n(x)$  可写为

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega_{n+1}(x) \quad (6.2.17)$$



由于式 (6.2.15)、式 (6.2.16)、式 (6.2.17) 的表达形式清晰简单, 在进行公式推导时, 显得更为方便。

② 对于插值多项式  $L_n(x)$ , 它除了满足插值条件外, 与  $f(x)$  再没有任何联系。而对于余项  $R_n(x)$ , 它除了与  $f(x)$  本身有联系外, 它还与  $f(x)$  的  $n+1$  阶导数  $f^{(n+1)}(\xi)$  有联系, 所以余项  $R_n(x)$  与  $f(x)$  的联系比插值函数  $L_n(x)$  与  $f(x)$  的联系紧密。

③ 由余项公式 (6.2.10) 可知, 当函数  $f(x)$  是不超过  $n$  次的多项式时, 由于  $f^{(n+1)}(\xi) = 0$ , 从而  $R_n(x) = 0$ , 因此  $f(x) \equiv L_n(x)$ 。这说明, 如果  $f(x)$  是不超过  $n$  次的多项式, 那么取  $n+1$  个互异节点的  $n$  次插值多项式就一定是它本身。特别地, 当  $f(x) \equiv 1$  时, 得恒等式  $\sum_{i=0}^n l_i(x) \equiv 1$ 。

④ 根据定理 6.1.1 知, 满足插值条件式 (6.1.1) 的插值多项式存在且唯一, 因此, 余项定理 6.2.1 不仅对拉格朗日插值多项式成立, 而且对满足式 (6.1.1) 的任何其他形式的插值多项式也恒成立。

⑤ 余项定理 6.2.1 只有在  $f(x)$  的高阶导数存在时才能应用, 且从估计式 (6.2.12) 可以看出,  $|R_n(x)|$  的大小除了与  $M_{n+1}$  有关外, 还与因子  $\left| \prod_{j=0}^n (x-x_j) \right|$  有关, 为了使  $R_n(x)$  尽量小, 显然要尽可能地选取靠近插值点  $x$  的若干个插值节点。

⑥ 如果函数  $f(x)$  只是以函数表的形式给出的, 而没有具体的解析式, 则无法估计  $|f^{(n+1)}(x)|$  的上界  $M_{n+1}$ , 从而不能直接应用式 (6.2.12)。这时, 就必须用事后误差估计的方法来估计误差, 所谓事后误差估计, 就是直接用计算结果来估计误差的方法。

例如, 有 3 个插值节点  $x_0, x_1, x_2$ , 可首先用  $x_0, x_1$  进行线性插值, 可求出  $f(x)$  的一个插值函数  $L_1(x)$ , 再用  $x_0, x_2$  进行线性插值, 求出  $f(x)$  的另一个插值函数  $\bar{L}_1(x)$ , 由余项式 (6.2.10) 得

$$f(x) - L_1(x) = \frac{f''(\xi_1)}{2}(x-x_0)(x-x_1)$$

$$f(x) - \bar{L}_1(x) = \frac{f''(\xi_2)}{2}(x-x_0)(x-x_2)$$

若  $f''(x)$  在插值区间  $[a, b]$  内变化不大, 可认为  $f''(\xi_1) \approx f''(\xi_2)$ , 于是将上述两式相除得

$$\frac{f(x) - L_1(x)}{f(x) - \bar{L}_1(x)} \approx \frac{x - x_1}{x - x_2}$$

简化整理后得

$$|f(x) - L_1(x)| \approx \left| \frac{x - x_1}{x_1 - x_2} (L_1(x) - \bar{L}_1(x)) \right| \leq \left| \frac{x - x_1}{x_1 - x_2} \right| |L_1(x) - \bar{L}_1(x)| \quad (6.2.18)$$

**例 6.2.8** 已知:  $\sqrt{100} = 10$ ,  $\sqrt{121} = 11$ ,  $\sqrt{144} = 12$ , 用事后误差估计的方法估计例 6.2.1 中  $\sqrt{115}$  近似值的误差。

**解** 取  $x_0 = 100, y_0 = 10, x_2 = 144, y_2 = 12$ , 构造线性插值多项式

$$\bar{L}_1(x) = \frac{x-144}{100-144} \times 10 + \frac{x-100}{144-100} \times 12$$

所以 
$$\bar{L}_1(115) = \frac{115-144}{100-144} \times 10 + \frac{115-100}{144-100} \times 12 = 10.6818$$

于是利用式 (6.2.18), 得

$$|f(115) - L_1(115)| \leq \left| \frac{115-121}{121-144} \right| \times |10.71428 - 10.6818| = 0.0084 \leq 0.05$$

故例 6.2.1 中  $\sqrt{115}$  的近似值 10.714 具有 3 位有效数字。

⑦ 尽管抛物插值的精度比线性插值好, 但并不能认为插值多项式的次数  $n$  越高越好。因为当  $n$  较大时, 经常有数值不稳定的现象, 特别当  $n \rightarrow \infty$  时, 插值函数  $L_n(x)$  并不一定收敛到被插函数  $f(x)$ 。所以, 一般取  $n \leq 7$ , 否则应该采用分段低次插值的方法 (见 6.5 节)。

## 6.3 牛顿插值

由直线方程两点式的推广而得到的拉格朗日插值公式形式对称、便于实现, 但是由于公式中的基函数  $l_i(x) (i=0,1,2,\dots,n)$  都依赖于全部插值节点, 因此, 在增加或减少节点时, 必须全部重新计算, 这样就增加了计算工作量。本节将从直线方程的点斜式出发, 推广出另一种具有递推形式的插值公式, 即牛顿插值公式。

假设  $f(x_0)=y_0, f(x_1)=y_1$ , 构造线性插值函数  $N_1(x)$ , 使

$$N_1(x_0)=y_0, \quad N_1(x_1)=y_1$$

由直线方程的点斜式可得

$$N_1(x)=y_0+\frac{y_1-y_0}{x_1-x_0}(x-x_0)$$

令

$$a_1=\frac{y_1-y_0}{x_1-x_0}$$

则

$$N_1(x)=y_0+a_1(x-x_0) \quad (6.3.1)$$

现将式 (6.3.1) 推广到 3 个点  $(x_0, y_0), (x_1, y_1), (x_2, y_2)$  的情况, 则可构造二次插值函数  $N_2(x)$ 。

设

$$N_2(x)=a_0+a_1(x-x_0)+a_2(x-x_0)(x-x_1) \quad (6.3.2)$$

$$\text{由插值条件} \begin{cases} N_2(x_0)=y_0 \text{ 得 } a_0=y_0 \\ N_2(x_1)=y_1 \text{ 得 } a_1=\frac{y_1-y_0}{x_1-x_0} \\ N_2(x_2)=y_2 \text{ 得 } a_2=\frac{\frac{y_2-y_0}{x_2-x_0}-\frac{y_1-y_0}{x_1-x_0}}{x_2-x_1} \end{cases}$$

我们可以将式 (6.3.2) 推广到  $n+1$  个点  $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$  的情况, 设

$$N_n(x)=a_0+a_1(x-x_0)+a_2(x-x_0)(x-x_1)+\dots+a_n(x-x_0)\cdots(x-x_{n-1}) \quad (6.3.3)$$

式中  $a_0, a_1, \dots, a_n$  为待定系数, 利用插值条件

$$N_n(x_i)=y_i \quad (i=0,1,\dots,n) \quad (6.3.4)$$

可求出  $a_i (i=0,1,2,\dots,n)$  的值, 从而求出  $n$  次插值多项式。但由于  $a_i (i=0,1,2,\dots,n)$  的值随着  $n$  的增大, 变得越来越复杂, 为了简化插值公式, 有必要引入差商 (也称均差) 的概念。

**定义 6.3.1 (差商)** 设  $f(x)$  在互异节点  $x_0, x_1, \dots, x_n$  上的函数值为  $f(x_0), f(x_1), \dots, f(x_n)$ , 称

$$f(x_0, x_1)=\frac{f(x_1)-f(x_0)}{x_1-x_0}$$

为  $f(x)$  关于  $x_0, x_1$  的一阶差商；称

$$f(x_0, x_1, x_2) = \frac{f(x_1, x_2) - f(x_0, x_1)}{x_2 - x_0}$$

为  $f(x)$  关于  $x_0, x_1, x_2$  的二阶差商；一般地，称

$$f(x_0, x_1, \dots, x_n) = \frac{f(x_1, x_2, \dots, x_n) - f(x_0, x_1, \dots, x_{n-1})}{x_n - x_0} \quad (6.3.5)$$

为  $f(x)$  关于  $x_0, x_1, \dots, x_n$  的  $n$  阶差商。

特别地，补充定义函数值  $f(x_i)$  为  $f(x)$  关于  $x_i$  的零阶差商。

差商具有如下性质。

① 差商对称性， $k$  阶差商可表示为函数值  $f(x_0), f(x_1), \dots, f(x_k)$  的线性组合，即

$$f(x_0, x_1, \dots, x_k) = \sum_{i=0}^k \frac{f(x_i)}{\prod_{\substack{j=0 \\ j \neq i}}^k (x_i - x_j)} \quad (6.3.6)$$

式 (6.3.6) 表明差商  $f(x_0, x_1, \dots, x_k)$  与节点排列次序无关。

**证明** 用数学归纳法证明。当  $k=1$  时，有

$$f(x_0, x_1) = \frac{f(x_0) - f(x_1)}{x_0 - x_1} = \frac{f(x_0)}{x_0 - x_1} + \frac{f(x_1)}{x_1 - x_0}$$

所以式 (6.3.6) 成立。

设  $k=m-1$  时，式 (6.3.6) 成立，即有

$$f(x_0, x_1, \dots, x_{m-1}) = \sum_{j=0}^{m-1} \frac{f(x_j)}{(x_j - x_0)(x_j - x_1) \cdots (x_j - x_{j-1})(x_j - x_{j+1}) \cdots (x_j - x_{m-1})}$$

和

$$f(x_1, x_2, \dots, x_m) = \sum_{j=1}^m \frac{f(x_j)}{(x_j - x_1)(x_j - x_2) \cdots (x_j - x_{j-1})(x_j - x_{j+1}) \cdots (x_j - x_m)}$$

由  $m$  阶差商的定义式 (6.3.5) 及归纳假设得

$$\begin{aligned} f(x_0, x_1, \dots, x_m) &= \frac{1}{x_0 - x_m} [f(x_0, x_1, \dots, x_{m-1}) - f(x_1, x_2, \dots, x_m)] \\ &= \frac{f(x_0)}{(x_0 - x_1)(x_0 - x_2) \cdots (x_0 - x_{m-1})} \cdot \frac{1}{x_0 - x_m} + \\ &\quad \sum_{j=1}^{m-1} \frac{f(x_j) \left( \frac{1}{x_j - x_0} - \frac{1}{x_j - x_m} \right)}{(x_j - x_1)(x_j - x_2) \cdots (x_j - x_{j-1})(x_j - x_{j+1}) \cdots (x_j - x_{m-1})} \cdot \frac{1}{x_0 - x_m} + \\ &\quad \frac{f(x_m)}{(x_m - x_1)(x_m - x_2) \cdots (x_m - x_{m-1})} \cdot \frac{1}{x_m - x_0} \\ &= \sum_{j=0}^m \frac{f(x_j)}{(x_j - x_0)(x_j - x_1) \cdots (x_j - x_{j-1})(x_j - x_{j+1}) \cdots (x_j - x_m)} \\ &= f(x_0, x_1, \dots, x_k) = \sum_{i=0}^k \frac{f(x_i)}{\prod_{\substack{j=0 \\ j \neq i}}^k (x_i - x_j)} \end{aligned}$$

证毕。

② 如果  $f(x, x_0, x_1, \dots, x_k)$  是  $x$  的  $m$  次多项式, 则  $f(x, x_0, \dots, x_k, x_{k+1})$  是  $x$  的  $(m-1)$  次多项式。事实上, 由于

$$f(x, x_0, \dots, x_k, x_{k+1}) = \frac{f(x, x_0, \dots, x_k) - f(x_0, x_1, \dots, x_{k+1})}{x - x_{k+1}}$$

右端分子为  $x$  的  $m$  次多项式, 且当  $x = x_{k+1}$  时, 分子为 0, 所以分子含有  $(x - x_{k+1})$  的因子, 与分母相约后得到  $(m-1)$  次多项式。

③ 若  $f(x) \in C^n[a, b]$ , 且  $x_i \in [a, b], (i = 0, 1, \dots, n)$  互异, 则有

$$f(x_0, x_1, \dots, x_n) = \frac{f^{(n)}(\xi)}{n!} \quad (\xi \in (a, b)) \quad (6.3.7)$$

此性质可由罗尔 (Rolle) 定理证明。

下面推导牛顿插值多项式。

根据差商定义 6.3.1, 把  $x$  看成在  $[a, b]$  内一点, 可得

$$\begin{aligned} f(x) &= f(x_0) + f(x, x_0)(x - x_0) \\ f(x, x_0) &= f(x_0, x_1) + f(x, x_0, x_1)(x - x_1) \\ &\dots \\ f(x, x_0, \dots, x_{n-1}) &= f(x_0, x_1, \dots, x_n) + f(x, x_0, \dots, x_n)(x - x_n) \end{aligned}$$

只要依次把后一式代入前一式, 就得到

$$\begin{aligned} f(x) &= f(x_0) + f(x_0, x_1)(x - x_0) + f(x_0, x_1, x_2)(x - x_0)(x - x_1) + \dots + \\ &\quad f(x_0, x_1, \dots, x_n)(x - x_0)(x - x_1) \cdots (x - x_{n-1}) + f(x, x_0, \dots, x_n)\omega_{n+1}(x) \\ &= N_n(x) + R_n(x) \end{aligned}$$

式中

$$N_n(x) = f(x_0) + f(x_0, x_1)(x - x_0) + \dots + f(x_0, x_1, \dots, x_n)(x - x_0)(x - x_1) \cdots (x - x_{n-1}) \quad (6.3.8)$$

$$R_n(x) = f(x) - N_n(x) = f(x, x_0, \dots, x_n)\omega_{n+1}(x) \quad (6.3.9)$$

这里,  $\omega_{n+1}(x) = (x - x_0)(x - x_1) \cdots (x - x_n)$ 。

由于  $\omega_{n+1}(x_i) = 0 (i = 0, 1, 2, \dots, n)$ , 因此  $R_n(x_i) = 0$ , 于是  $N_n(x_i) = f(x_i) (i = 0, 1, \dots, n)$  满足插值条件, 且次数不超过  $n$ 。因此式 (6.3.8) 称为  $f(x)$  的不超过  $n$  次的牛顿插值多项式。式 (6.3.9) 称为插值余项, 由插值多项式的唯一性可知, 它与式 (6.2.10) 是等价的, 但式 (6.3.9) 更具有一般性, 它对  $f(x)$  是由离散点给出的情形或  $f(x)$  导数不存在时均适用。牛顿插值公式 (6.3.8) 是  $n$  次插值多项式的又一种构造形式, 但它克服了拉格朗日插值多项式的缺点。它的一个明显的优点是, 每增加一个插值节点, 只要在原牛顿插值公式中增添一项就可形成高一次的插值公式, 因此, 它具有递推性质

$$N_{n+1}(x) = N_n(x) + (x - x_0)(x - x_1) \cdots (x - x_n)f(x_0, x_1, \dots, x_{n+1}) \quad (6.3.10)$$

另外, 从式 (6.3.8) 中可以看出, 牛顿插值公式中各项的系数就是函数  $f(x)$  的各阶差商  $f(x_0), f(x_0, x_1), \dots, f(x_0, x_1, \dots, x_n)$ , 因此, 在构造牛顿插值公式时, 常常先把差商列成一个表, 此表称为差商表 (见表 6.3.1)。

表 6.3.1 各阶差商表

$k$	$x_k$	$f(x_k)$	一阶差商	二阶差商	三阶差商	...
0	$x_0$	$f(x_0)$				...
1	$x_1$	$f(x_1)$	$f(x_0, x_1)$			...
2	$x_2$	$f(x_2)$	$f(x_1, x_2)$	$f(x_0, x_1, x_2)$		...

续表

$k$	$x_k$	$f(x_k)$	一阶差商	二阶差商	三阶差商	...
3	$x_3$	$f(x_3)$	$f(x_2, x_3)$	$f(x_1, x_2, x_3)$	$f(x_0, x_1, x_2, x_3)$	...
4	$x_4$	$f(x_4)$	$f(x_3, x_4)$	$f(x_2, x_3, x_4)$	$f(x_1, x_2, x_3, x_4)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

例 6.3.1 已知:  $\sqrt{100} = 10$ ,  $\sqrt{121} = 11$ ,  $\sqrt{144} = 12$ , 试用线性插值和抛物插值求  $\sqrt{115}$  的近似值。

解 首先构造差商表 (见表 6.3.2)。

表 6.3.2 例 6.3.1 的差商表

$x$	$\sqrt{x}$	一阶差商	二阶差商
100	10		
		0.047 619	
121	11		-0.000 094 11
		0.043 478	
144	12		

利用牛顿插值公式 (6.3.8) 线性插值得

$$\sqrt{115} \approx N_1(115) = 10 + 0.047\,619 \times (115 - 100) = 10.7143$$

用抛物插值得

$$\sqrt{115} \approx N_2(115) = N_1(115) + (-0.00009411)(115 - 100)(115 - 121) = 10.7143 + 0.0085 = 10.7228$$

从例 6.3.1 的计算过程可以看出, 与拉格朗日插值多项式相比较, 牛顿插值多项式的优点是明显的。

例 6.3.2 已知  $x = 1, 2, 3, 4, 5$  对应的函数值为  $f(x) = 1, 4, 7, 8, 6$ , 构造 4 次牛顿插值多项式, 并求  $f(1.5)$  近似值。

解 首先构造差商表 (见表 6.3.3)。

表 6.3.3 例 6.3.2 的差商表

$k$	$x_k$	$f(x_k)$	一阶差商	二阶差商	三阶差商	四阶差商
0	1	<u>1</u>				
			<u>3</u>			
1	2	4		<u>0</u>		
			3		<u>-1/3</u>	
2	3	7		-1		<u>1/24</u>
			1		-1/6	
3	4	8		-3/2		
			-2			
4	5	6				

以表 6.3.2 中带下画线的数字为系数, 利用式 (6.3.8) 直接可得

$$\begin{aligned}
N_4(x) &= 1 + 3(x-1) + 0(x-1)(x-2) + \\
&\quad \left(-\frac{1}{3}\right)(x-1)(x-2)(x-3) + \\
&\quad \frac{1}{24}(x-1)(x-2)(x-3)(x-4) \\
&= \dots \\
&= \frac{1}{24}x^4 - \frac{3}{4}x^3 + \frac{83}{24}x^2 - \frac{11}{4}x + 1 \\
f(1.5) &\approx N_4(1.5) = \frac{299}{128} \approx 2.3359375
\end{aligned}$$

此结果与例 6.2.3 的结果一致，这进一步说明了插值多项式的唯一性。

## 6.4 埃尔米特插值

许多实际问题不仅要求插值函数在节点上与原来的函数相等（满足插值条件），而且还要求在节点上的各阶导数值也相等。满足这些条件的插值，称为埃尔米特（Hermite）插值。

本节主要讨论已知两个节点  $x_0, x_1$  的函数值  $f(x_0)=y_0, f(x_1)=y_1$  和一阶导数值  $f'(x_0)=m_0, f'(x_1)=m_1$  的情形。

已知函数在两个互异节点  $x_0, x_1$  上的函数值  $f(x_0)=y_0, f(x_1)=y_1$  和一阶导数值  $f'(x_0)=m_0, f'(x_1)=m_1$ ，求一个三次插值多项式  $H(x)$ ，使其满足下式

$$\begin{cases} H(x_0) = y_0, & H(x_1) = y_1 \\ H'(x_0) = m_0, & H'(x_1) = m_1 \end{cases} \quad (6.4.1)$$

这样的  $H(x)$  称为两点三次埃尔米特插值多项式。

仍然采用构造插值基函数的方法，可设

$$H(x) = h_0(x)y_0 + h_1(x)y_1 + H_0(x)m_0 + H_1(x)m_1 \quad (6.4.2)$$

式中  $h_0(x), h_1(x), H_0(x), H_1(x)$  都为插值基函数，它们的取值见表 6.4.1。

表 6.4.1 各插值基函数的取值

基函数	函数值		一阶导数值	
	$x_0$	$x_1$	$x_0$	$x_1$
$h_0(x)$	1	0	0	0
$h_1(x)$	0	1	0	0
$H_0(x)$	0	0	1	0
$H_1(x)$	0	0	0	1

先求  $h_0(x)$ ，由于  $h_0(x_1)=h'_0(x_1)=0$ ，所以  $h_0(x)$  中必有因式  $(x-x_1)^2$ ，另外， $h_0(x)$  最多是一个三次多项式，因此可设

$$h_0(x) = (a + b(x-x_0)) \left( \frac{x-x_1}{x_0-x_1} \right)^2$$

利用  $h_0(x_0)=1$  得  $a=1$ , 为确定  $b$ , 对  $h_0(x)$  求导数, 再利用  $h'_0(x_0)=0$  得  $b=\frac{-2}{x_0-x_1}$ , 于是得

$$h_0(x)=\left(1+2\frac{x-x_0}{x_1-x_0}\right)\left(\frac{x-x_1}{x_0-x_1}\right)^2 \quad (6.4.3)$$

由对称性, 将式 (6.4.3) 中的  $x_0, x_1$  互换可得

$$h_1(x)=\left(1+2\frac{x-x_1}{x_0-x_1}\right)\left(\frac{x-x_0}{x_1-x_0}\right)^2 \quad (6.4.4)$$

接下来求  $H_0(x)$ , 由于  $H_0(x_0)=H_0(x_1)=0$  且  $H'_0(x_1)=0$ , 故  $H_0(x)$  中必有因式  $(x-x_0)(x-x_1)^2$ , 另外,  $H_0(x)$  是一个不超过三次的多项式, 于是可设

$$H_0(x)=a(x-x_0)\left(\frac{x-x_1}{x_0-x_1}\right)^2$$

式中,  $a$  是常数。为确定  $a$ , 对  $H_0(x)$  求导数, 再利用  $H'_0(x_0)=1$ , 可得  $a=1$ , 于是得

$$H_0(x)=(x-x_0)\left(\frac{x-x_1}{x_0-x_1}\right)^2 \quad (6.4.5)$$

由对称性, 将式 (6.4.5) 中的  $x_0, x_1$  互换可得

$$H_1(x)=(x-x_1)\left(\frac{x-x_0}{x_1-x_0}\right)^2 \quad (6.4.6)$$

将上述 4 个基函数  $h_0(x), h_1(x), H_0(x), H_1(x)$  代入式 (6.4.2), 即可求出  $H(x)$ 。容易验证  $H(x)$  满足式 (6.4.1)。

**例 6.4.1** 已知  $f(0)=0, f(1)=1, f'(0)=3, f'(1)=9$ , 构造三次 Hermite 插值  $H(x)$ 。

**解** 方法 1 (基函数法): 由式 (6.4.3) 到式 (6.4.6), 求基函数

$$h_0(x)=\left(1+2\frac{x-0}{1-0}\right)\left(\frac{x-1}{0-1}\right)^2=(1+2x)(1-x)^2$$

$$h_1(x)=\left(1+2\frac{x-1}{0-1}\right)\left(\frac{x-0}{1-0}\right)^2=(3-2x)x^2$$

$$H_0(x)=(x-0)\left(\frac{x-1}{0-1}\right)^2=x(1-x)^2$$

$$H_1(x)=(x-1)\left(\frac{x-0}{1-0}\right)^2=(x-1)x^2$$

于是

$$\begin{aligned} H(x) &= (1+2x)(1-x)^2 \times 0 + (3-2x)x^2 \times 1 + x(1-x)^2 \times 3 + (x-1)x^2 \times 9 \\ &= (3-2x)x^2 + 3x(1-x)^2 + 9x^2(x-1) \\ &= 10x^3 - 12x^2 + 3x \end{aligned}$$

方法 2 (待定系数法): 设

$$H(x)=a_0+a_1x+a_2x^2+a_3x^3$$

则

$$H'(x)=a_1+2a_2x+3a_3x^2$$

由所给插值条件得

$$\begin{aligned}0 &= H(0) = a_0 \\3 &= H'(0) = a_1 \\1 &= H(1) = a_0 + a_1 + a_2 + a_3 \\9 &= H'(1) = a_1 + 2a_2 + 3a_3\end{aligned}$$

解此方程组得

$$a_0 = 0, a_1 = 3, a_2 = -12, a_3 = 10$$

于是

$$H(x) = 10x^3 - 12x^2 + 3x$$

**定理 6.4.1 (余项定理)** 设  $H(x)$  是关于  $x_0, x_1$  两点三次 Hermite 插值, 若  $f(x) \in C^3[a, b], f^{(4)}(x)$  在  $[a, b]$  内存在, 其中,  $[a, b]$  是包含  $x_0, x_1$  的任意区间, 则对任意给定的  $x \in [a, b]$ , 总存在一点  $\xi$  (依赖于  $x$ ) 使

$$R(x) = f(x) - H(x) = \frac{f^{(4)}(\xi)}{4!} (x - x_0)^2 (x - x_1)^2 \quad (6.4.7)$$

**证明** 对于任意一个固定点  $x (x \neq x_0, x \neq x_1)$ , 引进辅助函数

$$\varphi(t) = f(t) - H(t) - \frac{R(x)}{(x - x_0)^2 (x - x_1)^2} (t - x_0)^2 (t - x_1)^2$$

显然,  $\varphi(t)$  具有四阶连续导数, 并且有  $x_0, x_1$  和  $x$  三个零点, 其中  $x_0, x_1$  是二重零点. 根据罗尔定理,  $\varphi'(t)$  在  $x_0, x_1$  和  $x$  构成的两个子区间内至少各有一个零点, 分别设为  $\xi_0$  和  $\xi_1$ , 这样,  $\varphi'(t)$  共有 4 个零点  $x_0, \xi_0, x_1, \xi_1$ . 反复利用罗尔定理可以推出  $\varphi^{(4)}(t)$  在  $[a, b]$  内至少有一个零点, 设为  $\xi$ , 对  $\varphi(t)$  求四阶导数, 并注意  $H(t)$  是三次多项式, 故

$$\varphi^{(4)}(t) = f^{(4)}(t) - 4! \frac{R(x)}{(x - x_0)^2 (x - x_1)^2}$$

将  $\xi$  代入上式, 并利用  $\varphi^{(4)}(\xi) = 0$ , 即得

$$R(x) = \frac{f^{(4)}(\xi)}{4!} (x - x_0)^2 (x - x_1)^2 \quad (a < \xi < b)$$

## 6.5 分段低次插值

### 6.5.1 高次插值与龙格现象

在插值方法中, 为了提高插值多项式的逼近程度, 常常需要增加节点的个数, 这样, 插值多项式  $L_n(x)$  的次数将逐次提高, 但是高次插值的逼近效果往往并不理想. 节点的增多固然使插值函数  $L_n(x)$  在更多的地方与  $f(x)$  相等, 但在两个插值节点之间,  $L_n(x)$  并不一定很好地逼近  $f(x)$ , 有时差异很大, 即高次插值的收敛性得不到保证. 另一方面, 从舍入误差的观点看, 高次插值由于计算量增大, 可能会产生严重的误差积累. 因此, 稳定性也得不到保证.

1901 年, 龙格 (Runge) 对函数  $f(x) = \frac{1}{1+x^2} (-5 \leq x \leq 5)$  取等距节点

$$x_k = -5 + kh \quad (h = \frac{10}{10}, k = 0, 1, \dots, 10)$$

求出拉格朗日插值多项式



$$L_n(x) = \sum_{k=0}^n \left( \prod_{\substack{i=0 \\ i \neq k}}^n \frac{x-x_i}{x_k-x_i} \right) \frac{1}{1+x_k^2} \quad (n=10) \quad (6.5.1)$$

如图 6.5.1 所示为  $L_{10}(x)$  和  $f(x) = \frac{1}{1+x^2}$  的函数图形（龙格现象）。

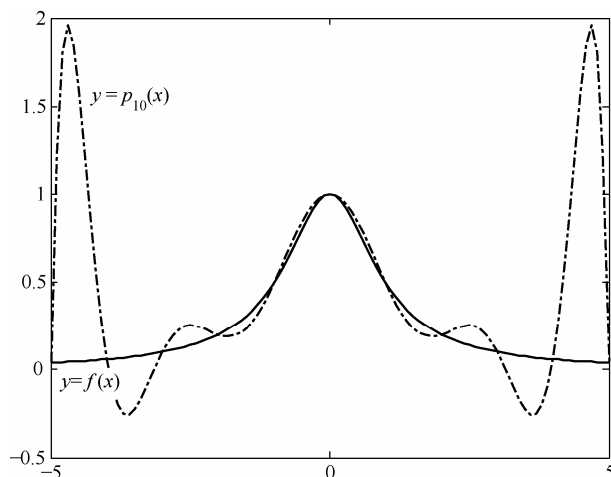


图 6.5.1 龙格现象

从图 6.5.1 可以看出，在接近区间两端点附近， $L_{10}(x)$  与  $f(x)$  的偏差很大，例如：

$$L_{10}(\pm 4.8) = 1.80438, \text{ 而 } f(\pm 4.8) = 0.4160$$

龙格还进一步证明了，在节点等距的条件下，当  $|x| \leq 3.63$  时，由式 (6.5.1) 计算的插值多项式  $L_n(x)$  满足

$$\lim_{n \rightarrow \infty} L_n(x) = f(x)$$

而当  $|x| \geq 3.63$  时， $\{L_n(x)\}$  发散。后来，人们把高次插值不收敛的现象称作龙格现象。

龙格现象说明，节点加密或大范围内使用高次插值不一定能保证插值函数能很好地逼近  $f(x)$ 。不过，如果在每个小区间  $[x_i, x_{i+1}]$  内应用低次插值，则可避免龙格现象的发生。实践证明，用一个分段低次插值多项式去逼近  $f(x)$ ，其效果要优于用一个任意光滑的高次插值多项式去逼近  $f(x)$ 。

## 6.5.2 分段线性插值

**定义 6.5.1（分段线性插值函数）** 设  $f(x)$  在给定的  $n+1$  个互异节点上的函数值为： $f(x_i) = y_i (i=0, 1, 2, \dots, n)$ ， $S_1(x)$  为区间  $[a, b]$  内的函数，如果  $S_1(x)$  满足下列条件：

- ①  $S_1(x)$  是在区间  $[a, b]$  内的每个小区间  $[x_i, x_{i+1}]$  ( $i=0, 1, 2, \dots, n$ ) 内的线性函数。
- ②  $S_1(x_i) = y_i (i=0, 1, 2, \dots, n)$ 。
- ③  $S_1(x)$  是插值区间  $[a, b]$  内的连续函数。

则称  $S_1(x)$  是插值区间  $[a, b]$  内的分段线性插值函数。

分段线性插值的几何意义是，在每个小区间  $[x_i, x_{i+1}]$  ( $i=0, 1, 2, \dots, n$ ) 内作连接插值点  $(x_i, y_i)$  与  $(x_{i+1}, y_{i+1})$  的直线。

函数  $S_1(x)$  在区间  $[x_{i-1}, x_i]$  ( $i=1, 2, \dots, n$ ) 内的表达式为

$$S_1(x) = \frac{x-x_i}{x_{i-1}-x_i} y_{i-1} + \frac{x-x_{i-1}}{x_i-x_{i-1}} y_i = l_{i-1}(x)y_{i-1} + l_i(x)y_i \quad (x_{i-1} \leq x \leq x_i)$$

函数  $S_1(x)$  在区间  $[x_i, x_{i+1}]$  ( $i=0, 1, 2, \dots, n-1$ ) 内的表达式为

$$S_1(x) = \frac{x-x_{i+1}}{x_i-x_{i+1}} y_i + \frac{x-x_i}{x_{i+1}-x_i} y_{i+1} = l_i(x)y_i + l_{i+1}(x)y_{i+1} \quad (x_i \leq x \leq x_{i+1})$$

如果用基函数表示, 则  $S_1(x)$  在插值区间  $[a, b]$  内的表达式可以统一表示为

$$S_1(x) = \sum_{i=0}^n l_i(x)y_i \quad (6.5.2)$$

式中

$$l_i(x) = \begin{cases} \frac{x-x_{i-1}}{x_i-x_{i-1}} & x_{i-1} \leq x \leq x_i \quad (i=0 \text{ 略去}) \\ \frac{x-x_{i+1}}{x_i-x_{i+1}} & x_i \leq x \leq x_{i+1} \quad (i=n \text{ 略去}) \\ 0 & x \in [a, b] - [x_{i-1}, x_{i+1}] \end{cases} \quad (6.5.3)$$

分段线性插值基函数  $l_i(x)$  只在  $x_i$  附近不为 0, 在其他地方均为 0, 这种性质称为局部非零性质。当插值点有误差时, 这种局部非零性质将误差控制在一个局部区域内。

分段线性插值多项式的余项可以通过线性插值多项式的余项估计。

**定理 6.5.1** 设  $f(x)$  在给定的互异节点  $x_i$  上的函数值为  $f(x_i) = y_i$  ( $i=0, 1, 2, \dots, n$ ),  $f(x) \in C^1[a, b]$ ,  $f''(x)$  在  $[a, b]$  内存在,  $S_1(x)$  是插值区间  $[a, b]$  内的由数据  $(x_i, y_i)$  ( $i=0, 1, 2, \dots, n$ ) 构成的分段线性插值函数, 则

$$|R(x)| = |f(x) - S_1(x)| \leq \frac{h^2}{8} M \quad (6.5.4)$$

式中,  $h = \max_{0 \leq i \leq n-1} |x_{i+1} - x_i|$ ,  $M = \max_{a \leq x \leq b} |f''(x)|$ 。

**证明** 由式 (6.2.13), 在每个小区间  $[x_i, x_{i+1}]$  ( $i=0, 1, 2, \dots, n$ ) 内有

$$|R(x)| \leq \frac{(x_{i+1} - x_i)^2}{8} \times \max_{x_i \leq x \leq x_{i+1}} |f''(x)| = R_i(x)$$

因此, 在整个区间  $[a, b]$  内有

$$|R(x)| \leq \max_{0 \leq i \leq n-1} |R_i(x)| \leq \frac{h^2}{8} M$$

证毕。

**例 6.5.1** 设  $f(x) = \frac{1}{1+x^2}$  在  $[-5, 5]$  内取  $n=10$ , 按等距节点求分段线性插值函数  $S_1(x)$ , 计算各相邻节点的中点处  $S_1(x)$  与  $f(x)$  的值, 并估计误差。

**解** 步长  $h = \frac{5-(-5)}{10} = 1$ ,  $x_i = -5 + ih = -5 + i$  ( $0 \leq i \leq 10$ ), 在区间  $[x_i, x_{i+1}]$  ( $i=0, 1, 2, \dots, 9$ ) 内的线性插值函数为

$$S_1^{(i)}(x) = \frac{x-x_{i+1}}{x_i-x_{i+1}} y_i + \frac{x-x_i}{x_{i+1}-x_i} y_{i+1} = \frac{x_{i+1}-x}{1+x_i^2} + \frac{x-x_i}{1+x_{i+1}^2} \quad (i=0, 1, 2, \dots, 9)$$

由分段线性插值函数的定义得

$$S_1(x) = S_1^{(i)}(x) = \frac{x_{i+1} - x}{1 + x_i^2} + \frac{x - x_i}{1 + x_{i+1}^2}, \quad x \in [x_i, x_{i+1}]$$

各小区间  $[x_i, x_{i+1}]$  ( $i = 0, 1, 2, \dots, 9$ ) 的中点的函数值及插值函数值见表 6.5.1, 函数图如图 6.5.2 所示。

表 6.5.1 中点的函数值及插值函数值

$x$	$\pm 0.5$	$\pm 1.5$	$\pm 2.5$	$\pm 3.5$	$\pm 4.5$
$f(x)$	0.800 00	0.307 69	0.137 93	0.075 47	0.047 06
$S_1(x)$	0.750 00	0.350 00	0.150 00	0.079 41	0.048 64

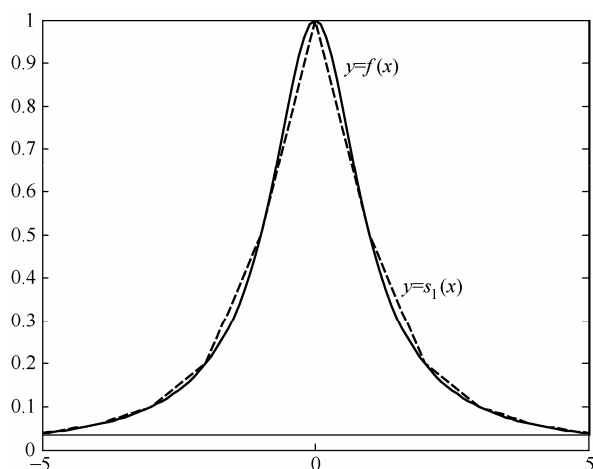


图 6.5.2 分段线性插值

由于  $f(x) = \frac{1}{1+x^2}$ , 则

$$f'(x) = \frac{2x}{(1+x^2)^2}, f''(x) = \frac{6x^2-2}{(1+x^2)^3}, f'''(x) = \frac{24x(1-x^2)}{(1+x^2)^4}$$

令

$$f'''(x) = \frac{24x(1-x^2)}{(1+x^2)^4} = 0$$

得  $f''(x)$  的极值点为: 0 和  $\pm 1$

于是

$$\max_{-5 \leq x \leq 5} \{|f''(x)|\} = \max \{|f''(0)|, |f''(\pm 1)|, |f''(\pm 5)|\} = 2$$

所以

$$|R(x)| \leq \frac{1}{8} \times 2 = 0.25 \quad (x \in [-5, 5])$$

### 6.5.3 分段三次埃尔米特插值

分段线性插值, 计算简单, 但光滑性差, 而分段三次埃尔米特插值则是一种光滑的分段插值。

**定义 6.5.2 (分段三次埃尔米特插值函数)** 设  $f(x)$  在给定的  $n+1$  个互异节点上的函数值

为:  $f(x_i) = y_i$  ( $i = 0, 1, 2, \dots, n$ ), 一阶导数值为:  $f'(x_i) = y'_i = m_i$  ( $i = 0, 1, 2, \dots, n$ ),  $S_3(x)$  为插值区间  $[a, b]$  内的函数, 若  $S_3(x)$  满足以下条件:

- (1)  $S_3(x)$  在每个小区间  $[x_i, x_{i+1}]$  ( $i = 0, 1, 2, \dots, n$ ) 内是三次多项式;
- (2)  $S_3(x_i) = y_i, S'_3(x_i) = m_i$  ( $i = 0, 1, 2, \dots, n$ );
- (3)  $S_3(x)$  在区间  $[a, b]$  内为一阶导数连续的函数。

则称  $S_3(x)$  是插值区间  $[a, b]$  内的分段三次埃尔米特插值函数。

利用 6.4 节的知识, 不难得出  $S_3(x)$  的表达式

$$S_3(x) = \sum_{i=0}^n \{h_i(x)y_i + H_i(x)m_i\} \quad (6.5.5)$$

式中,  $h_i(x), H_i(x)$  为插值基函数, 表达式为

$$h_i(x) = \begin{cases} \left(1 + 2 \frac{x - x_i}{x_{i-1} - x_i}\right) \left(\frac{x - x_{i-1}}{x_i - x_{i-1}}\right)^2 & x_{i-1} \leq x \leq x_i \quad (i = 0 \text{ 略去}) \\ \left(1 + 2 \frac{x - x_i}{x_{i+1} - x_i}\right) \left(\frac{x - x_{i+1}}{x_i - x_{i+1}}\right)^2 & x_i < x \leq x_{i+1} \quad (i = n \text{ 略去}) \\ 0 & x \in [a, b] - [x_{i-1}, x_{i+1}] \end{cases} \quad (6.5.6)$$

$$H_i(x) = \begin{cases} (x - x_i) \left(\frac{x - x_{i-1}}{x_i - x_{i-1}}\right)^2 & x_{i-1} \leq x \leq x_i \quad (i = 0 \text{ 略去}) \\ (x - x_i) \left(\frac{x - x_{i+1}}{x_i - x_{i+1}}\right)^2 & x_i < x \leq x_{i+1} \quad (i = n \text{ 略去}) \\ 0 & x \in [a, b] - [x_{i-1}, x_{i+1}] \end{cases} \quad (6.5.7)$$

分段三次埃尔米特插值余项可以通过两点三次埃尔米特多项式的插值余项得到。

**定理 6.5.2** 设  $S_3(x)$  是  $a = x_0 < x_1 < \dots < x_n = b$  内的分段三次埃尔米特插值函数,  $f(x) \in C^3[a, b]$ ,  $f^{(4)}(x)$  在  $[a, b]$  内存在, 对给定的  $x \in [a, b]$ , 总存在一点  $\xi \in (a, b)$ , 使

$$|R(x)| = |f(x) - S_3(x)| \leq \frac{h^4}{384} M_4 \quad (6.5.8)$$

式中

$$h = \max_{0 \leq i \leq n-1} |x_{i+1} - x_i|, M_4 = \max_{a \leq x \leq b} |f^{(4)}(x)|$$

**证明** 在每个小区间  $[x_i, x_{i+1}]$  ( $i = 0, 1, 2, \dots, n$ ) 内, 根据式 (6.4.7) 有

$$R(x) = \frac{f^{(4)}(\xi_i)}{4!} (x - x_i)^2 (x - x_{i+1})^2 \quad (x_i < \xi_i < x_{i+1})$$

由于

$$\max_{x_i \leq x \leq x_{i+1}} \{(x - x_i)^2 (x - x_{i+1})^2\} = \frac{(x_{i+1} - x_i)^4}{16}$$

因此

$$\max_{x_i \leq x \leq x_{i+1}} |R(x)| \leq \frac{(x_{i+1} - x_i)^4}{384} \max_{x_i \leq x \leq x_{i+1}} |f^{(4)}(x)|$$

于是, 在区间  $[a, b]$  内有

$$|R(x)| \leq \frac{h^4}{384} M_4$$

证毕。

**例 6.5.2** 给定函数  $f(x) = \frac{1}{1+x^2}$ ，函数值和一阶导数值见表 6.5.2。

表 6.5.2 例 6.5.2 的函数值和一阶导数值

$i$	0	1	2	3	4	5
$x_i$	0.0000	1.0000	2.0000	3.0000	4.0000	5.0000
$f(x_i)$	1.0000	0.500 00	0.200 00	0.100 00	0.058 82	0.038 46
$f'(x_i)$	0.0000	-0.500 00	-0.160 00	-0.060 00	-0.027 68	-0.014 79

用分段三次埃尔米特插值多项式计算  $f(0.5)$ ,  $f(1.5)$ ,  $f(2.5)$ ,  $f(3.5)$ ,  $f(4.8)$  的近似值并与准确值进行比较。

**解** 利用式 (6.5.5)、式 (6.5.6)、式 (6.5.7) 和式 (6.5.8) 计算，其结果见表 6.5.3。

表 6.5.3 例 6.5.2 的计算结果

$x$	$i$	$h_i(x)$	$H_i(x)$	$S_3(x)$	$f(x)$	$R(x)$
0.5	0	0.5	0.125	0.8125	0.8	0.0125
	1	0.5	-0.125			
1.5	1	0.5	0.125	0.3075	0.3077	-0.0002
	2	0.5	-0.125			
2.5	2	0.5	0.125	0.1375	0.1379	0.0004
	3	0.5	-0.125			
3.5	3	0.5	0.125	0.07537	0.07547	0.0010
	4	0.5	-0.125			
4.8	4	0.104	0.032	0.04158	0.04160	-0.0002
	5	0.896	-0.128			

从计算结果看，分段三次埃尔米特插值多项式逼近的效果是令人满意的。

## 6.6 样条函数插值

### 6.6.1 三次样条插值函数

6.5.3 节介绍的分段三次埃尔米特插值函数  $S_3(x)$ ，只有当被插函数在所有插值节点处的函数值和导数值都已知的条件下才能使用，而且  $S_3(x)$  在内节点处的二阶导数一般不连续。这是极不方便的。一方面，有时不可能也没有必要已知被插函数在内节点处的导数值；另一方面， $S_3(x)$  解决不了许多工程技术中提出的对插值函数的光滑性有较高要求的计算问题。例如，船体放样的形值线，高速飞机的机翼形线，内燃机的进、排气门的凸轮曲线，都要求曲线具有较高的光滑程度，不仅要连续，而且要有连续的曲率，即二阶导数连续。为解决这一类问题，导

致了样条插值的产生。

样条 (Spline) 的概念来源于生产实践, “样条” 是绘制曲线的一种绘图工具, 它是富有弹性的细长条。绘图时用压铁使样条通过指定的形值点 (样点), 并调整样点使它具有满意的形状, 然后沿样条画出曲线, 这种曲线称为样条曲线。它实际上是由分段三次曲线 “装配” 起来的, 在形值点处具有二阶连续导数, 由此抽象出的数学模型称为样条函数。

**定义 6.6.1 (三次样条插值函数)** 设在插值区间  $[a, b]$  内给出一组互异节点  $a \leq x_0 < x_1 < \cdots < x_n \leq b$ , 若函数  $S(x)$  满足以下条件:

①  $S(x) \in C^2[a, b]$ , 即  $S(x)$  在  $[a, b]$  内为二阶导数连续的函数;

②  $S(x)$  在每个小区间  $[x_i, x_{i+1}] (i = 0, 1, 2, \cdots, n-1)$  内是三次多项式。

则称  $S(x)$  是节点  $x_0, x_1, \cdots, x_n$  上的三次样条函数。

③ 若  $S(x)$  在节点上还满足插值条件:

$$S(x_i) = f(x_i) \quad (i = 0, 1, 2, \cdots, n) \quad (6.6.1)$$

则称  $S(x)$  为  $[a, b]$  内的三次样条插值函数。

由定义 6.6.1 可知, 三次样条插值函数  $S(x)$  是通过全部样点, 并且是二阶导数连续的分段三次插值函数。它在每个小区间  $[x_i, x_{i+1}]$  内是三次多项式, 有 4 个待定系数, 由于在插值区间  $[a, b]$  内共有  $n$  个小区间, 故  $S(x)$  有  $4n$  个待定系数, 而由定义 6.6.1 中的条件①可知,  $S(x)$  在  $(n-1)$  个内节点上应该满足

$$\begin{cases} S(x_i - 0) = S(x_i + 0) \\ S'(x_i - 0) = S'(x_i + 0) \\ S''(x_i - 0) = S''(x_i + 0) \end{cases} \quad (6.6.2)$$

它给出了  $3(n-1)$  个条件, 再加上式 (6.6.1) 给出的  $(n+1)$  个条件, 共  $(4n-2)$  个条件, 求解  $S(x)$  仍缺两个条件, 为此, 要根据问题要求补充两个边界条件。

$$\text{第一边界条件:} \quad S'(x_0) = f'_0, \quad S'(x_n) = f'_n \quad (6.6.3)$$

$$\text{第二边界条件:} \quad S''(x_0) = f''_0, \quad S''(x_n) = f''_n \quad (6.6.4)$$

特别地, 当  $S''(x_0) = S''(x_n) = 0$  时, 称为自然边界条件。

第三边界条件: 当  $f(x)$  为周期函数时, 因为  $f(x_0) = f(x_n)$ , 此时

$$S(x_0) = S(x_n) = f(x_0) \quad \text{并且} \quad S'(x_0 + 0) = S'(x_n - 0), S''(x_0 + 0) = S''(x_n - 0)$$

这时,  $S(x)$  成为周期样条函数。

**例 6.6.1** 已知函数  $f(x)$  在三个点处的值为  $f(-1) = 1, f(0) = 0, f(1) = 1$ , 在区间  $[-1, 1]$  内, 求  $f(x)$  在自然边界条件下的三次样条插值函数  $S(x)$ 。

**解** 将区间  $[-1, 1]$  分成两个子区间  $[-1, 0]$  和  $[0, 1]$ , 故设

$$S(x) = \begin{cases} S_0(x) = a_0x^3 + b_0x^2 + c_0x + d_0, & x \in [-1, 0] \\ S_1(x) = a_1x^3 + b_1x^2 + c_1x + d_1, & x \in [0, 1] \end{cases}$$

由插值条件:  $S_0(-1) = 1, S_0(0) = 0, S_1(0) = 0, S_1(1) = 1$ , 得

$$\begin{cases} -a_0 + b_0 - c_0 = 1 \\ d_0 = 0 \\ d_1 = 0 \\ a_1 + b_1 + c_1 = 1 \end{cases} \quad (6.6.5)$$

在内节点  $x = 0$  处,  $S'(x), S''(x)$  连续, 并且  $S'_0(0) = S'_1(0), S''_0(0) = S''_1(0)$ , 得

$$\begin{cases} c_0 = c_1 \\ b_0 = b_1 \end{cases} \quad (6.6.6)$$

最后, 由边界条件  $S_0''(-1) = 0, S_1''(1) = 0$ , 得

$$\begin{cases} -6a_0 + 2b_0 = 0 \\ 6a_1 + 2b_1 = 0 \end{cases} \quad (6.6.7)$$

联立式 (6.6.5)、式 (6.6.6) 和式 (6.6.7), 求解关于待定系数的线性方程组, 得

$$a_0 = -a_1 = \frac{1}{2}, b_0 = b_1 = \frac{3}{2}, c_0 = c_1 = d_0 = d_1 = 0$$

从而, 问题的解为

$$S(x) = \begin{cases} S_0(x) = \frac{1}{2}x^3 + \frac{3}{2}x^2 & x \in [-1, 0] \\ S_1(x) = -\frac{1}{2}x^3 + \frac{3}{2}x^2 & x \in [0, 1] \end{cases}$$

这种解法称为待定系数法, 当  $n$  较大时, 由于要解  $4n$  阶的线性方程组, 工作量太大, 因此, 在实际计算时一般不采用待定系数法, 而考虑其他比较简单的方法。

## 6.6.2 三次样条插值函数的求法

设  $S(x)$  在节点  $x_i$  处的一阶导数为  $m_i$ , 即  $S'(x_i) = m_i$  ( $i = 0, 1, 2, \dots, n$ ), 因为  $S(x)$  在每个小区间  $[x_i, x_{i+1}]$  内都是三次多项式, 因此, 可以将  $S(x)$  表示成整个区间内的分段两点三次埃尔米特插值多项式, 当  $x \in [x_i, x_{i+1}]$  且  $h_i = x_{i+1} - x_i$  时, 有

$$\begin{aligned} S(x) &= \left(1 + 2 \frac{x - x_i}{h_i}\right) \left(\frac{x - x_{i+1}}{-h_i}\right)^2 y_i + \left(1 + 2 \frac{x - x_{i+1}}{-h_i}\right) \left(\frac{x - x_i}{h_i}\right)^2 y_{i+1} + \\ &\quad (x - x_i) \left(\frac{x - x_{i+1}}{-h_i}\right)^2 m_i + (x - x_{i+1}) \left(\frac{x - x_i}{h_i}\right)^2 m_{i+1} \\ &= \frac{[h_i + 2(x - x_i)](x - x_{i+1})^2}{h_i^3} y_i + \frac{[h_i - 2(x - x_{i+1})](x - x_i)^2}{h_i^3} y_{i+1} + \\ &\quad \frac{(x - x_i)(x - x_{i+1})^2}{h_i^2} m_i + \frac{(x - x_{i+1})(x - x_i)^2}{h_i^2} m_{i+1} \end{aligned} \quad (6.6.8)$$

对  $S(x)$  求二次导数, 并整理得

$$\begin{aligned} S''(x) &= \frac{6x - 2x_i - 4x_{i+1}}{h_i^2} m_i + \frac{6x - 4x_i - 2x_{i+1}}{h_i^2} m_{i+1} + \\ &\quad \frac{6(x_i + x_{i+1} - 2x)}{h_i^3} (y_{i+1} - y_i) \quad (x \in [x_i, x_{i+1}]) \end{aligned} \quad (6.6.9)$$

于是

$$\lim_{x \rightarrow x_i + 0} S''(x) = -\frac{4}{h_i} m_i - \frac{2}{h_i} m_{i+1} + \frac{6}{h_i^2} (y_{i+1} - y_i)$$

在式 (6.6.8) 中以  $i-1$  取代  $i$ , 以  $i$  取代  $i+1$ , 可得  $S(x)$  在  $[x_{i-1}, x_i]$  内的表达式, 然后可得

$$\lim_{x \rightarrow x_i - 0} S''(x) = \frac{2}{h_{i-1}} m_{i-1} + \frac{4}{h_{i-1}} m_i - \frac{6}{h_{i-1}^2} (y_i - y_{i-1})$$

由  $\lim_{x \rightarrow x_i+0} S''(x) = \lim_{x \rightarrow x_i-0} S''(x)$ , 得

$$\frac{1}{h_{i-1}} m_{i-1} + 2 \left( \frac{1}{h_{i-1}} + \frac{1}{h_i} \right) m_i + \frac{1}{h_i} m_{i+1} = 3 \left( \frac{y_{i+1} - y_i}{h_i^2} + \frac{y_i - y_{i-1}}{h_{i-1}^2} \right) \quad (6.6.10)$$

( $i = 1, 2, \dots, n-1$ )

用  $\frac{1}{h_{i-1}} + \frac{1}{h_i}$  即  $\frac{h_i + h_{i-1}}{h_{i-1} h_i}$  除式 (6.6.10) 两边, 并化简所得方程, 得

$$\lambda_i m_{i-1} + 2m_i + \mu_i m_{i+1} = d_i \quad (i = 1, 2, \dots, n-1) \quad (6.6.11)$$

式中

$$\lambda_i = \frac{h_i}{h_i + h_{i-1}}, \quad \mu_i = \frac{h_{i-1}}{h_i + h_{i-1}}$$

$$d_i = 3 \left( \mu_i \frac{y_{i+1} - y_i}{h_i} + \lambda_i \frac{y_i - y_{i-1}}{h_{i-1}} \right)$$

当  $i$  取遍  $1, 2, \dots, n-1$  时, 得到含有  $(n-1)$  个方程及  $(n+1)$  个未知数  $m_0, m_1, \dots, m_n$  的方程组 (6.6.11), 其中每个方程都含有  $S(x)$  在相邻三个节点上的一阶导数值。节点  $x_i$  处的一阶导数  $m_i$ , 在力学上的意义为细梁在  $x_i$  截面处的转角, 因此式 (6.6.11) 也称为三转角方程。为了确定未知数  $m_i$  ( $i = 0, 1, 2, \dots, n$ ), 还需要补充两个边界条件, 即可得到关于  $m_i$  的三对角方程组。下面就两种边界条件, 分别进行讨论。

① 如果问题要求  $S(x)$  满足第一边界条件, 此时

$$m_0 = f'_0, \quad m_n = f'_n$$

于是, 可将方程组 (6.6.11) 化为  $n-1$  阶方程组

$$\begin{pmatrix} 2 & \mu_1 & & & \\ \lambda_2 & 2 & \mu_2 & & \\ & \lambda_3 & 2 & \mu_3 & \\ & & \ddots & \ddots & \ddots \\ & & & \lambda_{n-2} & 2 & \mu_{n-2} \\ & & & & \lambda_{n-1} & 2 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ \vdots \\ m_{n-2} \\ m_{n-1} \end{pmatrix} = \begin{pmatrix} d_1 - \lambda_1 f'_0 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-2} \\ d_{n-1} - \mu_{n-1} f'_n \end{pmatrix} \quad (6.6.12)$$

② 如果问题要求  $S(x)$  满足第二边界条件, 此时可以利用第二边界条件  $S''(x_0) = f''_0, S''(x_n) = f''_n$ , 分别在  $[x_0, x_1]$  和  $[x_{n-1}, x_n]$  内建立关于  $m_0, m_1$  和  $m_{n-1}, m_n$  的方程。

在式 (6.6.9) 中, 令  $i = 0, x = x_0$ , 得

$$S''(x_0) = -\frac{4}{h_0} m_0 - \frac{2}{h_0} m_1 + \frac{6}{h_0^2} (y_1 - y_0) = f''_0$$

从而有

$$2m_0 + m_1 = 3 \frac{y_1 - y_0}{h_0} - \frac{h_0}{2} f''_0 \quad (6.6.13)$$

在式 (6.6.9) 中, 令  $i = n-1, x = x_n$ , 得

$$S''(x_n) = \frac{2}{h_{n-1}} m_{n-1} + \frac{4}{h_{n-1}} m_n - \frac{6}{h_{n-1}^2} (y_n - y_{n-1}) = f''_n$$

从而有

$$m_{n-1} + 2m_n = 3 \frac{y_n - y_{n-1}}{h_{n-1}} - \frac{h_{n-1}}{2} f''_n \quad (6.6.14)$$



将式 (6.6.13)、式 (6.6.14) 与式 (6.6.11) 联立, 得  $n+1$  阶方程组

$$\begin{pmatrix} 2 & 1 & & & \\ \lambda_1 & 2 & \mu_1 & & \\ & \lambda_2 & 2 & \mu_2 & \\ & & \ddots & \ddots & \ddots \\ & & & \lambda_{n-1} & 2 & \mu_{n-1} \\ & & & & 1 & 2 \end{pmatrix} \begin{pmatrix} m_0 \\ m_1 \\ m_2 \\ \vdots \\ m_{n-1} \\ m_n \end{pmatrix} = \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix} \quad (6.6.15)$$

式中  $d_i$  ( $i=1,2,\dots,n-1$ ) 与式 (6.6.11) 一致, 而  $d_0, d_n$  的值为

$$d_0 = 3 \frac{y_1 - y_0}{h_0} - \frac{h_0}{2} f_0'', \quad d_n = 3 \frac{y_n - y_{n-1}}{h_{n-1}} + \frac{h_{n-1}}{2} f_n'' \quad (6.6.16)$$

由式 (6.6.11) 知,  $\lambda_i + \mu_i = 1$  ( $i=1,2,\dots,n-1$ ), 且  $\lambda_i, \mu_i$  都为正数, 所以方程组 (6.6.12) 和方程组 (6.6.15) 的系数矩阵都是严格对角占优的三对角矩阵, 可以用追赶法求其唯一的解。

**例 6.6.2** 已知函数表 (见表 6.6.1), 求满足边界条件  $y'(0)=17/8, y'(5)=-19/8$  的三次样条插值函数, 并求  $f(3)$  和  $f(4.5)$  的近似值。

表 6.6.1 例 6.6.2 的函数表

$i$	0	1	2	3
$x_i$	1	2	4	5
$f(x_i)$	1	3	4	2

**解** 由表 6.6.1 知,  $h_0 = x_1 - x_0 = 1, h_1 = 2, h_2 = 1$ , 按式 (6.6.11) 计算得

$$\lambda_1 = 2/3, \lambda_2 = 1/3, \mu_1 = 1/3, \mu_2 = 2/3, d_1 = 9/2, d_2 = -7/2$$

将其代入方程组 (6.6.12) 得

$$\begin{cases} \frac{3}{2}m_0 + 2m_1 + \frac{1}{3}m_2 = \frac{9}{2} \\ \frac{1}{3}m_1 + 2m_2 + \frac{2}{3}m_3 = \frac{-7}{2} \end{cases}$$

由于

$$m_0 = f'_0 = 17/8, m_3 = f'_3 = -19/8$$

于是有

$$\begin{cases} 2m_1 + \frac{1}{3}m_2 = \frac{37}{12} \\ \frac{1}{3}m_1 + 2m_2 = \frac{-23}{12} \end{cases}$$

其解为

$$m_1 = \frac{7}{4}, m_2 = -\frac{5}{4}$$

将其代入式 (6.6.8), 当  $i=0$  时, 得

$$\begin{aligned} S_0(x) &= \left[ (1+2(x-1))(x-2)^2 \right] \times 1 + \left[ (1-2(x-2))(x-1)^2 \right] \times 3 + \\ &\quad (x-1)(x-2)^2 \times \frac{17}{8} + (x-2)(x-1)^2 \times \frac{7}{4} \quad (x \in [1, 2]) \\ &= -\frac{1}{8}x^3 + \frac{3}{8}x^2 + \frac{7}{4}x - 1 \end{aligned}$$

同理, 可求得在其他子区间内的表达式为

$$S_1(x) = -\frac{1}{8}x^3 + \frac{3}{8}x^2 + \frac{7}{4}x - 1 \quad (x \in [2, 4])$$

$$S_2(x) = \frac{3}{8}x^3 - \frac{45}{8}x^2 + \frac{103}{4}x - 33 \quad (x \in [4, 5])$$

于是, 所求三次样条插值函数的分段表达式为

$$S(x) = \begin{cases} S_0(x) = -\frac{1}{8}x^3 + \frac{3}{8}x^2 + \frac{7}{4}x - 1 & (x \in [1, 4]) \\ S_2(x) = \frac{3}{8}x^3 - \frac{45}{8}x^2 + \frac{103}{4}x - 33 & (x \in [4, 5]) \end{cases}$$

并由此可求得

$$f(3) \approx S_0(3) = -\frac{1}{8} \times 3^3 + \frac{3}{8} \times 3^2 + \frac{7}{4} \times 3 - 1 = \frac{17}{4}$$

$$f(4.5) \approx S_2(4.5) = \frac{3}{8} \times 4.5^3 - \frac{45}{8} \times 4.5^2 + \frac{103}{4} \times 4.5 - 33 = \frac{201}{64}$$

可以证明, 对于三次样条插值函数来说, 当插值节点逐渐加密时, 不但样条函数收敛于被插函数本身, 而且其导数也同样收敛到被插函数的导数, 这种性质要优于多项式插值。因而, 为了提高精度, 只需要加密分段节点, 而不需要提高样条函数的次数。三次样条插值有明确的力学背景: 样条曲线可以看做弹性细梁受到集中载荷作用而生成的绕度曲线, 在扰动不大的情况下, 这种绕度曲线在数学上恰好表现为三次样条函数, 集中载荷的作用点就是三次样条函数的节点。

## 本章小结

本章介绍了复杂函数的多项式插值。拉格朗日插值适合求固定节点的函数值, 牛顿插值可以在计算过程中根据精度要求逐步增加节点且计算量小, 埃尔米特插值适合于导数值已知的情況。所有插值多项式的次数都不能太高, 否则会出现龙格现象。分段低次插值, 在缩小节点之间距离时, 能保证插值函数在整个区间内充分接近被插函数, 但光滑性差。样条函数光滑性好, 但构造和计算较复杂, 并且需要求解一个三对角方程组。

## 习题 6

6.1 给定离散点(1,0), (2,-5), (3,-6), (4,3), 试求拉格朗日插值多项式。

6.2 设  $x_0, x_1, \dots, x_n$  为  $n+1$  个互异节点,  $l_i(x) (i=0, 1, \dots, n)$  为拉格朗日插值基函数, 试证:

$$(1) \sum_{i=0}^n l_i(x) x_i^k \equiv x^k (k=0, 1, 2, \dots, n)$$

$$(2) \sum_{i=0}^n (x_i - x)^k l_i(x) \equiv 0 (k=0, 1, \dots, n)$$

6.3 设  $f(x) = \frac{1}{x}$ , 节点  $x_0 = 2, x_1 = 2.5, x_2 = 4$ , 求  $f(x)$  的抛物插值多项式  $L_2(x)$ , 计算  $f(3)$

的近似值并估计误差。

6.4 已知由数据(0,0), (0.5,y), (1,3)和(2,2)构造出的三次插值多项式  $p_3(x)$  的  $x^3$  的系数是 6, 试确定数据  $y$ 。

6.5 依据如下函数值表建立不超过 3 次的拉格朗日插值多项式及牛顿插值多项式, 并验证插值多项式的唯一性。

$x$	0	1	2	4
$f(x)$	1	9	23	3

6.6 证明: 拉格朗日插值函数

$$l_0(x) = \frac{(x-x_1)(x-x_2)\cdots(x-x_n)}{(x_0-x_1)(x_0-x_2)\cdots(x_0-x_n)}$$

可表示为如下牛顿形式

$$l_0(x) = 1 + \frac{(x-x_0)}{(x_0-x_1)} + \frac{(x-x_0)(x-x_1)}{(x_0-x_1)(x_0-x_2)} + \cdots + \frac{(x-x_0)(x-x_1)\cdots(x-x_{n-1})}{(x_0-x_1)(x_0-x_2)\cdots(x_0-x_n)}$$

6.7 证明: 当  $m > n$  时, 下式成立。

$$\sum_{i=0}^n (-1)^{n-i} \frac{c_m^n c_n^i}{m-i} = \frac{1}{m-n}$$

式中

$$c_m^n = \frac{m!}{n!(m-n)!}, \quad c_n^i = \frac{n!}{i!(n-i)!}$$

6.8 求  $f(x) = x^{n+1}$  关于节点  $x_0, x_1, \dots, x_n$  的拉格朗日插值多项式, 并利用插值余项定理证明下式。

$$\sum_{i=0}^n x_i^{n+1} l_i(0) = (-1)^n x_0 x_1 \cdots x_n$$

式中,  $l_i(x)$  为关于节点  $x_0, x_1, \dots, x_n$  的拉格朗日插值基函数。

6.9 给定如下数据表:

$x_i$	1	2	4	6	7
$f(x_i)$	4	1	0	1	1

求 4 次牛顿插值多项式, 并写出插值余项。

6.10 给定函数  $f(x) = x^3 - 4x$ , 试建立关于节点  $x_i = i+1 (i=0, 1, \dots, 5)$  的差商表, 并列出于节点  $x_0, x_1, x_2, x_3$  的插值多项式  $p(x)$ 。

6.11 设  $x_0, x_1, \dots, x_n$  点互异, 考察  $f(x) = \prod_{i=0}^n (x-x_i)$  的各阶差商, 证明:

$$f(x_0, x_1, \dots, x_k) = 0 \quad (k = 0, 1, \dots, n)$$

$$f(x_0, x_1, \dots, x_n, x) \equiv 1$$

6.12 设  $n, k$  为整数,  $0 \leq k \leq n-1$ , 证明:

$$\sum_{i=0}^n \frac{i^k}{\prod_{\substack{j=0 \\ j \neq i}}^n (i-j)} = 0$$

6.13 设首项系数为 1 的  $n$  次  $f(x)$  有  $n$  个互异的零点  $x_i (i=1,2,\cdots,n)$ , 证明:

$$\sum_{j=1}^n \frac{x_j^k}{f'(x_j)} = \begin{cases} 0 & k=0,1,\cdots,n-2 \\ 1 & k=n-1 \end{cases}$$

6.14 证明:

$$(1) 1+2+\cdots+n = \frac{1}{2}n(n+1)$$

$$(2) 1^3+2^3+\cdots+n^3 = \left[ \frac{n(n+1)}{2} \right]^2$$

$$(3) 1 \times 2 + 2 \times 3 + \cdots + n(n+1) = \frac{1}{3}n(n+1)(n+2)$$

$$(4) 1 \times 3 + 2 \times 4 + \cdots + n(n+2) = \frac{1}{6}n(n+1)(2n+7)$$

6.15 设节点  $x_i (i=1,2,\cdots,n)$  与点  $a$  互异, 试对  $f(x) = \frac{1}{a-x}$ , 证明:

$$f(x_0, x_1, \cdots, x_k) = \prod_{i=0}^k \frac{1}{a-x_i} \quad (k=0,1,\cdots,n)$$

并列  $f(x)$  的牛顿插值多项式。

6.16 依据以下数据表:

$x_i$	-2	-1	0	1	2	3
$y_i$	-5	1	1	1	7	25

构造出的插值多项式有多少次? 为什么? 试具体列出其插值多项式。

6.17 给定以下数据:

$x_i$	1	2
$f(x_i)$	2	3
$f'(x_i)$	0	-1

试构造埃尔米特插值多项式  $H_3(x)$  并计算  $f(1.5)$ 。

6.18 求次数小于等于 5 的多项式  $p(x)$ , 使满足以下插值条件:

$x_i$	0	1	2
$y_i$	2	1	2
$y'_i$	-2	-1	
$y''_i$	-10		

6.19 求  $f(x) = x^2$  在  $[a,b]$  内的分段线性插值函数  $I_h(x)$ , 并估计误差。

6.20 求  $f(x) = x^4$  在  $[a,b]$  内的分段三次埃尔米特插值函数  $I_h(x)$ , 并估计误差。

6.21 设分段式

$$S(x) = \begin{cases} x^3 + x^2 & 0 \leq x \leq 1 \\ 2x^3 + bx^2 + cx - 1 & 1 \leq x \leq 2 \end{cases}$$

是以 0, 1, 2 为节点的三次样条函数, 试确定系数  $b, c$  的值。

6.22 设  $f$  为定义在区间  $[0,3]$  内的函数, 部分的节点为  $x_i = 0+i$  ( $i=0,1,2,3$ ), 并给出  $f(x_0)=0, f(x_1)=0.5, f(x_2)=2.0, f(x_3)=1.5, y_i = f(x_i)$  ( $i=0,1,2,3$ )。

(1) 当  $y'_0 = f'(x_0)=0.2, y'_3 = f'(x_3)=-1$  时, 试求三次样条插值函数  $S(x)$ , 使其满足第一边界条件。

(2) 当  $y''_0 = f''(x_0)=-0.3, y''_3 = f''(x_3)=3.3$  时, 试求三次样条插值函数  $S(x)$ , 使其满足第二边界条件。

## 第7章 函数逼近



### 学习要点

函数逼近是指对于给定的复杂函数,构造一个简单函数使其整体误差最小,主要内容有:

- (1) 函数内积与正交多项式的概念。
- (2) 函数的一致逼近与最佳一致逼近多项式。
- (3) 函数的平方逼近与最佳平方逼近多项式。
- (4) 函数的离散曲线拟合多项式。



### 教学建议

本章为选学内容,正交多项式是数值计算的重要工具,可用 2 学时讲解,其他内容根据学时的多少,可有选择地进行取舍。建议学时 6~8 学时。

## 7.1 引言

函数逼近与函数插值既有相同点,又有不同点,相同点是二者都是对实际中的复杂函数构造一个既能反映函数本身的特性又便于计算的简单函数,近似替代原来的函数。不同点是,函数插值是对一组离散点  $(x_i, f(x_i)) (i=0,1,2,\dots,n)$ , 选定一个便于计算的简单函数形式  $p(x)$ , 要求简单函数  $p(x)$  满足  $p(x_i) = f(x_i) (i=0,1,2,\dots,n)$ , 由此确定函数  $p(x)$  作为  $f(x)$  的近似函数。而函数逼近是对实际中的复杂函数  $f(x)$ , 构造一个简单函数  $p(x)$ , 要求其误差从整体上在某种度量意义下最小。

以  $f(x)$  和  $p(x)$  的最大误差  $\|p-f\|_{\infty} = \max_{a \leq x \leq b} |p(x)-f(x)|$  (函数  $p(x)-f(x)$  的  $\infty$ -范数) 作为度量逼近函数  $p(x)$  和被逼近函数  $f(x)$  的逼近程度, 若存在一个函数序列  $p_n(x)$  满足  $\lim_{n \rightarrow \infty} \|p_n(x)-f(x)\|_{\infty} = 0$ , 则可证明  $\{p_n(x)\}$  在  $[a,b]$  内一致收敛到  $f(x)$ , 序列  $\{p_n(x)\}$  对  $f(x)$  的逼近称为一致逼近。

以积分  $\|p-f\|_2^2 = \int_a^b (p(x)-f(x))^2 dx$  (函数  $p(x)-f(x)$  的 2-范数) 作为度量逼近函数  $p(x)$  和被逼近函数  $f(x)$  的逼近程度, 若存在一个函数序列  $\{p_n(x)\}$  满足  $\lim_{n \rightarrow \infty} \|p_n(x)-f(x)\|_2^2 = 0$ , 则可证明  $\{p_n(x)\}$  在  $[a,b]$  内平方收敛到  $f(x)$ , 序列  $\{p_n(x)\}$  对  $f(x)$  的逼近称为平方逼近。

在实际计算中,在数值上求  $f(x)$  的一致逼近和平方逼近有一定的难度,我们更感兴趣的是求  $f(x)$  的最佳一致逼近和最佳平方逼近。

以离散节点的平方和  $\|p-f\|_2^2 = \sum_{i=0}^n (p(x_i) - f(x_i))^2$  (离散意义上的函数  $p(x) - f(x)$  的 2-范数平方) 作为度量逼近函数  $p(x)$  和被逼近函数  $f(x)$  的逼近程度,  $p(x)$  对  $f(x)$  的逼近称为离散平方逼近, 也称为离散曲线拟合。

## 7.2 函数的内积与正交多项式

### 7.2.1 权函数和函数的内积

**定义 7.2.1 (权函数)** 设  $[a, b]$  为有限或无限区间,  $\rho(x)$  是定义在  $[a, b]$  内的非负函数, 并且  $\int_a^b \rho(x)x^k dx$  对  $k=0, 1, 2, \dots$  都存在。对非负的  $f(x) \in C[a, b]$ , 若  $\int_a^b \rho(x)f(x)dx = 0$  时可得  $f(x) \equiv 0$ , 则称  $\rho(x)$  为  $[a, b]$  内的权函数。

权函数  $\rho(x)$  的作用主要是说明函数点  $x$  在  $[a, b]$  内所占据的重要性。

常见的权函数有:

$$\textcircled{1} \quad \rho(x) = 1 \quad (-1 \leq x \leq 1)$$

$$\textcircled{2} \quad \rho(x) = \frac{1}{\sqrt{1-x^2}} \quad (-1 \leq x \leq 1)$$

$$\textcircled{3} \quad \rho(x) = e^{-x} \quad (0 < x < +\infty)$$

$$\textcircled{4} \quad \rho(x) = e^{-x^2} \quad (-\infty < x < +\infty)$$

**定义 7.2.2 (连续区间上的函数内积)** 设  $f(x), g(x) \in [a, b]$ ,  $\rho(x)$  为  $[a, b]$  内的权函数, 则称

$$(f, g) = \int_a^b \rho(x)f(x)g(x)dx \quad (7.2.1)$$

为  $f(x)$ ,  $g(x)$  在区间  $[a, b]$  内的内积。

**定义 7.2.3 (离散节点上的函数内积)** 给定点集  $\{x_i\} (i=0, 1, 2, \dots, m)$  以及各点的权系数  $\{\omega_i\} (i=0, 1, 2, \dots, m)$ , 称

$$(f, g) = \sum_{i=0}^m \omega_i f(x_i)g(x_i) \quad (7.2.2)$$

为函数  $f(x)$ ,  $g(x)$  在离散节点  $\{x_i\} (i=0, 1, 2, \dots, m)$  上的内积。

不论是连续区间上的函数内积还是离散节点上的函数内积, 函数的内积都有如下性质:

$$\textcircled{1} \quad (f, g) = (g, f)$$

$$\textcircled{2} \quad (c_1 f + c_2 g, h) = c_1 (f, h) + c_2 (g, h)$$

$$\textcircled{3} \quad (f, f) \geq 0, \text{ 当且仅当 } f \equiv 0 \text{ 时, } (f, f) = 0$$

### 7.2.2 正交函数系

**定义 7.2.4 (连续区间上的函数正交)** 设  $f(x), g(x) \in [a, b]$ ,  $\rho(x)$  为  $[a, b]$  内的权函数, 若函数的内积

$$(f, g) = \int_a^b \rho(x)f(x)g(x)dx = 0$$

则称  $f(x)$ ,  $g(x)$  是在  $[a, b]$  内带权  $\rho(x)$  的正交。

**定义 7.2.5 (离散节点上的函数正交)** 给定点集  $\{x_i\} (i=0, 1, 2, \dots, m)$ , 以及各点的权系数  $\{\omega_i\} (i=0, 1, 2, \dots, m)$ , 若内积为

$$(f, g) = \sum_{i=0}^m \omega_i f(x_i) g(x_i) = 0$$

则称  $f(x)$ ,  $g(x)$  是在点集  $\{x_i\} (i=0, 1, 2, \dots, m)$  上带权  $\omega_i (i=0, 1, 2, \dots, m)$  的正交。

**定义 7.2.6 (连续区间上的正交函数系)** 设函数系  $\{\varphi_0(x), \varphi_1(x), \dots, \varphi_k(x), \dots\}$  是  $[a, b]$  内的连续函数。若满足条件

$$(\varphi_i, \varphi_j) = \int_a^b \rho(x) \varphi_i(x) \varphi_j(x) dx = \begin{cases} 0 & i \neq j \\ A_j > 0 & i = j \end{cases} \quad (i, j = 0, 1, 2, \dots)$$

则称函数系  $\{\varphi_k(x)\}$  是  $[a, b]$  内带权  $\rho(x)$  的正交函数系。

**定义 7.2.7 (离散节点上的正交函数系)** 设函数系  $\{\varphi_0(x), \varphi_1(x), \dots, \varphi_k(x), \dots\}$  是  $[a, b]$  内的连续函数。给定点集  $\{x_i\} (i=0, 1, 2, \dots, m)$  以及各点的权系数  $\{\omega_i\} (i=0, 1, 2, \dots, m)$ , 若满足条件

$$(\varphi_k, \varphi_j) = \sum_{i=0}^m \omega_i \varphi_k(x_i) \varphi_j(x_i) = \begin{cases} 0 & k \neq j \\ A_j > 0 & k = j \end{cases} \quad (k, j = 0, 1, 2, \dots)$$

则称函数系  $\{\varphi_k(x)\}$  是在给定点集  $\{x_i\} (i=0, 1, 2, \dots, m)$ , 上带权  $\omega_i (i=0, 1, 2, \dots, m)$  的正交函数系。

**例 7.2.1** 证明三角函数系  $\{1, \sin x, \cos x, \sin 2x, \cos 2x, \dots\}$  在  $[-\pi, \pi]$  内是正交函数系 ( $\rho(x) \equiv 1$ )。

**证明**

$$(1, 1) = \int_{-\pi}^{\pi} dx = 2\pi$$

$$(\sin nx, \sin mx) = \int_{-\pi}^{\pi} \sin nx \sin mx dx = \begin{cases} \pi & m = n \\ 0 & m \neq n \end{cases} \quad (n, m = 1, 2, \dots)$$

$$(\cos nx, \cos mx) = \int_{-\pi}^{\pi} \cos nx \cos mx dx = \begin{cases} \pi & m = n \\ 0 & m \neq n \end{cases} \quad (n, m = 1, 2, \dots)$$

$$(\cos nx, \sin mx) = \int_{-\pi}^{\pi} \cos nx \sin mx dx = 0 \quad (n, m = 0, 1, 2, \dots)$$

**定义 7.2.8 (连续区间上的正交多项式系)** 设  $p_n(x)$  是首项系数  $a_n \neq 0$  的  $n$  次多项式, 如果多项式序列  $\{p_n(x)\}_0^\infty$  满足

$$(p_i, p_j) = \int_a^b \rho(x) p_i(x) p_j(x) dx = \begin{cases} 0 & i \neq j \\ A_j > 0 & i = j \end{cases} \quad (7.2.3)$$

则称多项式序列  $\{p_n(x)\}_0^\infty$  为在  $[a, b]$  上带权  $\rho(x)$  的  $n$  次正交多项式系。

**定义 7.2.9 (离散节点上的正交多项式系)** 设  $p_n(x)$  是首项系数  $a_n \neq 0$  的  $n$  次多项式, 如果多项式序列  $\{p_n(x)\}_0^\infty$  满足

$$(p_k, p_j) = \sum_{i=0}^m \omega_i p_k(x_i) p_j(x_i) = \begin{cases} 0 & k \neq j \\ A_j > 0 & k = j \end{cases} \quad (k, j = 0, 1, 2, \dots) \quad (7.2.4)$$

则称多项式序列  $\{p_n(x)\}_0^\infty$  为在给定点集  $\{x_i\} (i=0, 1, 2, \dots, m)$  上带权  $\omega_i (i=0, 1, 2, \dots, m)$  的  $n$  次正交多项式系。

可以证明, 最高次项系数为 1 (首 1) 的正交多项式系  $\{p_k(x)\}$  有下面的递推公式



$$\begin{cases} p_0(x) = 1 \\ p_1(x) = (x - \alpha_0)p_0(x) = x - \alpha_0 \\ p_{k+1}(x) = (x - \alpha_k)p_k(x) - \beta_{k-1}p_{k-1}(x) \end{cases} \quad (7.2.5)$$

式中

$$\begin{cases} \alpha_k = \frac{(xp_k, p_k)}{(p_k, p_k)} & (k = 0, 1, 2, \dots, n-1) \\ \beta_{k-1} = \frac{(p_k, p_k)}{(p_{k-1}, p_{k-1})} & (k = 1, 2, \dots, n) \end{cases} \quad (7.2.6)$$

### 7.2.3 勒让德多项式

在区间  $[-1, 1]$  内, 权函数  $\rho(x) = 1$  的正交多项式, 称为勒让德 (Legendre) 多项式。其表达式为

$$\begin{aligned} P_0(x) &= 1 \\ P_n(x) &= \frac{1}{2^n n!} \frac{d^n}{dx^n} \left\{ (x^2 - 1)^n \right\} \quad (n = 1, 2, \dots) \end{aligned} \quad (7.2.7)$$

$P_n(x)$  的首项系数为  $\frac{(2n)!}{2^n (n!)^2}$ , 记

$$\tilde{P}_0(x) = 1, \quad \tilde{P}_n(x) = \frac{n!}{(2n)!} \frac{d^n}{dx^n} \left\{ (x^2 - 1)^n \right\} \quad (n = 1, 2, \dots) \quad (7.2.8)$$

则  $\tilde{P}_n(x)$  是首项系数为 1 的勒让德多项式。

勒让德多项式的性质如下。

① 正交性

$$(P_n, P_m) = \int_{-1}^1 P_n(x) P_m(x) dx = \begin{cases} 0 & m \neq n \\ \frac{2}{2n+1} & m = n \end{cases} \quad (7.2.9)$$

**证明** 令  $\varphi(x) = (x^2 - 1)^n$ , 则  $\varphi^{(k)}(\pm 1) = 0$  ( $k = 0, 1, 2, \dots, n-1$ ), 且  $P_n(x) = \frac{1}{2^n n!} \varphi^{(n)}(x)$ 。

设  $R(x)$  为不超过  $n$  次的多项式, 用分部积分得

$$\begin{aligned} \int_{-1}^1 P_n(x) R(x) dx &= \frac{1}{2^n n!} \int_{-1}^1 R(x) \varphi^{(n)}(x) dx = -\frac{1}{2^n n!} \int_{-1}^1 R^{(1)}(x) \varphi^{(n-1)}(x) dx \\ &= \dots = \frac{(-1)^n}{2^n n!} \int_{-1}^1 R^{(n)}(x) \varphi(x) dx \end{aligned}$$

当  $R(x)$  为不超过  $n-1$  次的多项式时,  $R^{(n)}(x) = 0$ , 于是有

$$\int_{-1}^1 P_n(x) P_m(x) dx = 0 \quad (m \neq n)$$

当  $R(x) = P_n(x)$  时, 则  $R^{(n)}(x) = P_n^{(n)}(x) = \frac{(2n)!}{2^n n!}$ , 于是

$$\int_{-1}^1 P_n^2(x) dx = \frac{(-1)^n (2n)!}{2^{2n} (n!)^2} \int_{-1}^1 (x^2 - 1)^n dx = \frac{2}{2n+1}$$

证毕。

② 递推公式

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x) \quad (n=1, 2, \dots) \quad (7.2.10)$$

证明 因为  $xP_n(x)$  为  $n+1$  次多项式, 所以可设

$$xP_n(x) = a_0P_0(x) + a_1P_1(x) + \dots + a_{n+1}P_{n+1}(x)$$

两边同乘  $P_k(x)$ , 并积分得

$$\int_{-1}^1 xP_n(x)P_k(x)dx = a_k \int_{-1}^1 P_k^2(x)dx$$

当  $k \leq n-2$  时,  $xP_k(x)$  的次数不超过  $n-1$ , 上式左边积分为 0, 因此  $a_k = 0$ 。当  $k = n$  时,  $xP_n^2(x)$  为奇函数, 上式左边积分仍为 0, 因此  $a_n = 0$ 。于是得

$$xP_n(x) = a_{n-1}P_{n-1}(x) + a_{n+1}P_{n+1}(x)$$

式中

$$\begin{aligned} a_{n-1} &= \frac{2n-1}{2} \int_{-1}^1 xP_n(x)P_{n-1}(x)dx = \frac{2n-1}{2} \cdot \frac{2n}{4n^2-1} = \frac{n}{2n+1} \\ a_{n+1} &= \frac{2n+3}{2} \int_{-1}^1 xP_n(x)P_{n+1}(x)dx = \frac{2n+3}{2} \cdot \frac{2(n+1)}{(2n+1)(2n+3)} = \frac{n+1}{2n+1} \end{aligned}$$

从而得式 (7.2.10), 证毕。

由  $P_0(x)=1$ ,  $P_1(x)=x$  并利用式 (7.2.10), 可得

$$P_2(x) = \frac{1}{2}(3x^2 - 1)$$

$$P_3(x) = \frac{1}{2}(5x^3 - 3x)$$

$$P_4(x) = \frac{1}{8}(35x^4 - 30x^2 + 3)$$

$$P_5(x) = \frac{1}{8}(63x^5 - 70x^3 + 15x)$$

.....

③ 奇偶性

$$P_n(-x) = (-1)^n P_n(x)$$

④  $P_n(x)$  在区间  $[-1, 1]$  内有  $n$  个不同的实零点。

## 7.2.4 切比雪夫多项式

在区间  $[-1, 1]$  内权函数  $\rho(x) = \frac{1}{\sqrt{1-x^2}}$  的正交多项式称为切比雪夫 (Chebyshev) 多项式, 可

表示为

$$T_n(x) = \cos(n \arccos x) \quad (n=0, 1, \dots) \quad (7.2.11)$$

若令  $x = \cos \theta$ , 则  $T_n(x) = \cos n\theta$  ( $0 \leq \theta \leq \pi$ ), 利用三角函数公式, 可将  $\cos n\theta$  展开成  $\cos \theta$  的一个  $n$  次多项式, 故式 (7.2.11) 为  $x$  的  $n$  次多项式。

切比雪夫多项式的性质如下。

① 正交性

$$(T_n, T_m) = \int_{-1}^1 \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}} dx = \begin{cases} 0 & m \neq n \\ \frac{\pi}{2} & m = n \neq 0 \\ \pi & m = n = 0 \end{cases} \quad (7.2.12)$$

事实上, 令  $x = \cos \theta$ ,  $dx = -\sin \theta d\theta$ , 代入式 (7.2.11), 得

$$(T_n, T_m) = \int_{-1}^1 \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}} dx = \int_0^\pi \cos n\theta \sin m\theta d\theta = \begin{cases} 0 & m \neq n \\ \frac{\pi}{2} & m = n \neq 0 \\ \pi & m = n = 0 \end{cases}$$

② 递推公式

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \quad (n=1, 2, \dots) \quad (7.2.13)$$

式中  $T_0(x) = 1$ ,  $T_1(x) = x$ 。

事实上, 由于  $x = \cos \theta$ ,  $T_{n+1}(x) = \cos(n+1)\theta$ , 利用三角函数公式  $\cos(n+1)\theta - \cos(n-1)\theta = 2\cos\theta\cos n\theta$ , 可得

$$\begin{aligned} T_2(x) &= 2x^2 - 1 \\ T_3(x) &= 4x^3 - 3x \\ T_4(x) &= 8x^4 - 8x^2 + 1 \\ T_5(x) &= 16x^5 - 20x^3 + 5x \\ T_6(x) &= 32x^6 - 48x^4 + 18x^2 - 1 \\ &\dots \end{aligned}$$

③ 奇偶性

$$T_n(-x) = (-1)^n T_n(x)$$

④  $T_n(x)$  在  $(-1, 1)$  内的  $n$  个零点为  $x_k = \cos \frac{2k-1}{2n}\pi$  ( $k=1, 2, \dots, n$ )

在  $[-1, 1]$  内有  $n+1$  个极值点,  $y_k = \cos \frac{k}{n}\pi$  ( $k=0, 1, 2, \dots, n$ )

⑤  $T_n(x)$  的最高次幂  $x^n$  的系数为  $2^{n-1} (n \geq 1)$ 。

## 7.3 最佳一致逼近

### 7.3.1 基本概念

**定义 7.3.1** (一致逼近多项式) 设  $f(x) \in [a, b]$ , 对任意给定的  $\varepsilon > 0$ , 若存在多项式  $p(x)$ , 使不等式  $\max_{a \leq x \leq b} |p(x) - f(x)| < \varepsilon$  成立, 则称多项式  $p(x)$  在  $[a, b]$  内一致逼近于  $f(x)$ 。

1885 年, Weierstrass 证明了一致逼近多项式的存在性。

**定理 7.3.1** (Weierstrass 定理) 设  $f(x) \in C[a, b]$ , 那么对于任意给定的  $\varepsilon > 0$ , 都存在多项式  $p(x)$ , 使得  $\max_{a \leq x \leq b} |p(x) - f(x)| < \varepsilon$ 。

1912 年, 伯恩斯坦 (Bernstein) 构造出了一致逼近多项式。

$$B_n(f, x) = \sum_{k=0}^n f\left(\frac{k}{n}\right) P_k(x)$$

式中  $P_k(x) = \binom{n}{k} x^k (1-x)^{n-k}$ ,  $\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{k!}$ , 且  $\lim_{n \rightarrow \infty} B_n(f, x) = f(x)$  在  $[0, 1]$  内一致成立。

在实际计算时, 一般先固定多项式的次数  $n$ , 再求一个多项式  $p_n^*(x)$  使  $\max_{a \leq x \leq b} |p_n^*(x) - f(x)|$  最小。这就是最佳一致逼近多项式。

**定义 7.3.2 (偏差)** 记  $P_n = \{\text{次数不超过 } n \text{ 的多项式的全体}\}$ , 设  $f(x) \in C[a, b]$ ,  $p(x) \in P_n$ , 记

$$\|f - p\|_{\infty} = \max_{a \leq x \leq b} |f(x) - p(x)| = \mu \quad (7.3.1)$$

称  $\mu$  为  $p(x)$  与  $f(x)$  的偏差。若  $\exists x_0 \in [a, b]$ , 使  $|f(x_0) - p(x_0)| = \mu$ , 则称  $x_0$  是  $p(x)$  关于  $f(x)$  的偏差点。若  $f(x_0) - p(x_0) = \mu$ , 则称  $x_0$  为正偏差点; 若  $f(x_0) - p(x_0) = -\mu$ , 则称  $x_0$  为负偏差点。

**定义 7.3.3 (最佳一致逼近多项式)** 设  $f(x) \in C[a, b]$ , 若存在  $p_n^*(x) \in P_n$ , 使

$$\|f - p_n^*\|_{\infty} = \min_{p \in P_n} \|f - p\|_{\infty} \quad (7.3.2)$$

则称  $p_n^*(x)$  为  $f(x)$  在  $[a, b]$  内的最佳一致逼近多项式。

**定理 7.3.2 (存在唯一性)** 若  $f(x) \in C[a, b]$ , 则存在唯一的  $p_n^*(x) \in P_n$ , 使得

$$\|f - p_n^*\|_{\infty} = \min_{p \in P_n} \|f - p\|_{\infty}$$

**定理 7.3.3 (切比雪夫定理)**  $p_n^*(x) \in P_n$  是  $f(x) \in C[a, b]$  的最佳一致逼近多项式的充要条件是: 在  $[a, b]$  内至少有  $n+2$  个轮流为正负的偏差点, 即: 至少有  $n+2$  个点  $a \leq x_1 < x_2 < \cdots < x_{n+2} \leq b$ , 使

$$p_n^*(x_k) - f(x_k) = (-1)^k \sigma \|f - p_n^*\|_{\infty} \quad (7.3.3)$$

式中,  $\sigma = \pm 1$ ,  $k = 1, 2, \dots, n+2$ 。

上述点  $\{x_k\}_{k=1}^{n+2}$  称为切比雪夫交错点组。

**例 7.3.1** 给定函数  $f(x) = 4x^3 + 2x^2 + x + 1$ , 试在区间  $[-1, 1]$  内求  $f(x)$  在  $P_2$  中的最佳一致逼近多项式。

**解** 由于 3 次切比雪夫多项式

$$T_3(x) = 4x^3 - 3x$$

在区间  $[-1, 1]$  内, 当  $x_k = \cos \frac{k\pi}{3}$  ( $k = 0, 1, 2$ ) 时, 轮流取最大值 1 和最小值 -1。如果将  $T_3(x)$  写成

$T_3(x) = 4x^3 - 3x = (4x^3 + 2x^2 + x + 1) - (2x^2 + 4x + 1) = f(x) - p_2(x)$ , 则  $x_k = \cos \frac{k\pi}{3}$  ( $k = 0, 1, 2$ ) 就是  $f(x) - p_2(x)$  的交错点组。也就是说,  $p_2(x) = 2x^2 + 4x + 1$  就是所求的二次最佳一致逼近多项式。

## 7.3.2 线性最佳一致逼近多项式

尽管定理 7.3.3 给出了最佳逼近多项式的基本特征和求最佳逼近多项式的主要依据, 但计

算最佳一致逼近多项式仍然十分困难, 本节只介绍线性最佳一致逼近多项式  $p_1(x)$  的求法。

**例 7.3.2** 设  $f(x)$  在  $[a, b]$  内有二阶导数, 且  $f''(x)$  在  $[a, b]$  内不变号, 求  $f(x)$  的线性最佳一致逼近多项式  $p_1(x) = a_0 + a_1x$ 。

**解** 由于  $f(x) \in C[a, b]$ , 且  $f''(x)$  在  $[a, b]$  内不变号, 由定理 7.3.3 可知, 存在点  $a \leq x_1 < x_2 < x_3 \leq b$ , 使

$$p_1(x_k) - f(x_k) = (-1)^k \sigma \max_{a \leq x \leq b} |p_1(x) - f(x)| \quad (k = 1, 2, 3, \sigma = \pm 1)$$

因为  $f''(x) \neq 0$ , 所以  $f'(x)$  在  $[a, b]$  内单调, 于是有  $f'(x) - p_1'(x) = f'(x) - a_1 = 0$ , 在  $[a, b]$  内只有一个根  $x_2$ , 故  $p_1(x)$  对  $f(x)$  另两个偏差点只能在  $[a, b]$  的端点。故有:  $x_1 = a, x_3 = b$ , 由此可得

$$p_1(a) - f(a) = -[p_1(x_2) - f(x_2)] = p_1(b) - f(b)$$

即

$$a_0 + a_1a - f(a) = a_0 + a_1b - f(b) \quad (7.3.4)$$

$$a_0 + a_1a - f(a) = -[a_0 + a_1x_2 - f(x_2)] \quad (7.3.5)$$

由式 (7.3.4) 可求得

$$a_1 = \frac{f(b) - f(a)}{b - a} \quad (7.3.6)$$

代入式 (7.3.5) 得

$$a_0 = \frac{1}{2}[f(a) - f(x_2)] - \frac{a + x_2}{2} \frac{f(b) - f(a)}{b - a} \quad (7.3.7)$$

式中,  $x_2$  由  $f'(x_2) = a_1$  求得, 于是可以求得  $p_1(x) = a_0 + a_1x$ 。

**例 7.3.3** 设  $f(x) = \sqrt{1+x^2}$ , 在  $[0, 1]$  内求  $f(x)$  的最佳一致逼近多项式  $p_1(x) = a_0 + a_1x$

**解** 由式 (7.3.6) 式得

$$a_1 = \frac{\sqrt{2} - 1}{1} \approx 0.414$$

由  $f'(x) = \frac{x}{\sqrt{1+x^2}}$  得

$$f'(x_2) = \frac{x_2}{\sqrt{1+x_2^2}} = a_1 \approx 0.414$$

所以  $x_2 = \sqrt{\frac{\sqrt{2}-2}{2}} = 0.4551$ ,  $f(x_2) = \sqrt{1+x_2^2} = 1.0986$ 。

$$a_0 = \frac{1}{2}(1 - 1.0986) - \frac{0.4551}{2} \times (\sqrt{2} - 1) = 0.955$$

所以  $f(x)$  的最佳一致逼近多项式为  $p_1(x) = 0.955 + 0.414x$ 。

**例 7.3.4** 求  $f(x) = \sqrt{x}$ , 在  $\left[\frac{1}{4}, 1\right]$  内的最佳一致逼近多项式  $p_1(x) = a_0 + a_1x$

**解**  $x_1 = \frac{1}{4}, x_3 = 1$ , 由式 (7.3.6) 得  $a_1 = \frac{\sqrt{1} - \sqrt{\frac{1}{4}}}{1 - \frac{1}{4}} = \frac{2}{3}$

由  $f'(x) = \frac{1}{2\sqrt{x}}$  得  $\frac{1}{2\sqrt{x_2}} = \frac{2}{3}$ ,  $x_2 = \frac{9}{16}$

由式 (7.3.7) 得  $a_0 = \frac{1}{2} \left( \frac{1}{2} + \frac{3}{4} - \frac{2}{3} \left( \frac{1}{4} + \frac{9}{16} \right) \right) = \frac{17}{48}$

所以  $p_1(x) = \frac{2}{3} + \frac{17}{48}x$  为  $f(x)$  在  $\left[\frac{1}{4}, 1\right]$  内的线性最佳一致逼近多项式。

### 7.3.3 近似最佳一致逼近多项式

在实际计算中, 求连续函数的最佳一致逼近多项式十分困难, 一种可行的方法是利用切比雪夫多项式的性质, 求近似最佳一致逼近多项式。由切比雪夫多项式的性质⑤可知,  $T_n(x)$  的最高次幂  $x^n$  的系数为  $2^{n-1} (n \geq 1)$ , 则  $\tilde{T}_n(x) = \frac{1}{2^{n-1}} T_n(x)$  是最高项系数为 1 的  $n$  次多项式。

**定理 7.3.4 (与零偏差最小的多项式)** 所有最高项系数为 1 的  $n$  次多项式中, 在区间  $[-1, 1]$  内与零偏差最小的多项式是  $\tilde{T}_n(x)$ 。

**证明** 由于  $\tilde{T}_n(x) = \frac{1}{2^{n-1}} T_n(x) = x^n - p_{n-1}^*(x)$ , 当  $x_k = \cos \frac{k}{n} \pi$  ( $k = 0, 1, 2, \dots, n$ ) 时, 有

$$\tilde{T}_n(x) = \frac{1}{2^{n-1}} (\cos n \arccos x_k) = \frac{1}{2^{n-1}} \cos k\pi = \frac{(-1)^k}{2^{n-1}} \quad (k = 0, 1, 2, \dots, n)$$

这说明  $p_{n-1}^*(x)$  与  $f(x) = x^n$  有  $n+1$  个轮流为正负的偏差点。根据定理 7.3.3 可知,  $p_{n-1}^*(x) \in P_{n-1}$  是  $f(x) = x^n$  的最佳一致逼近多项式, 即

$$\max_{-1 \leq x \leq 1} |\tilde{T}_n(x)| = \min_{p_{n-1}(x) \in P_{n-1}} \|x^n - p_{n-1}(x)\|_{\infty}$$

所以  $\tilde{T}_n(x)$  是在  $[-1, 1]$  内的与零偏差最小的多项式。证毕。

**例 7.3.5** 设  $f(x) = x^4$  在  $[-1, 1]$  内, 求  $f(x)$  在  $P_3$  中的最佳一致逼近多项式。

**解** 因为  $\tilde{T}_4(x) = \frac{1}{8} (8x^4 - 8x^2 + 1) = x^4 - (x^2 - \frac{1}{8})$

所以  $P_3^* = x^2 - \frac{1}{8}$

在实际应用中, 一种更常用和可行的方法是将  $f(x) \in C[-1, 1]$  直接按  $\{T_k(x)\}_0^{\infty}$  展开, 并用其部分和逼近  $f(x)$ , 由于  $\{T_k(x)\}_0^{\infty}$  是在区间  $[-1, 1]$  内带权  $\rho(x) = \sqrt{1-x^2}$  正交的多项式, 构造  $n$  次多项式

$$p_n^*(x) = \frac{a_0^*}{2} + \sum_{k=1}^n a_k^* T_k(x)$$

式中  $a_k^* = \frac{2}{\pi} \int_{-1}^1 \frac{f(x) T_k(x)}{\sqrt{1-x^2}} dx \quad (k = 0, 1, \dots, n)$

可以证明: 如果  $f(x) \in C[-1, 1]$ , 则  $\lim_{n \rightarrow \infty} \|f(x) - C_n^*(x)\|_{\infty} = 0$ , 由此可得

$$f(x) - p_n^*(x) \approx a_{n+1}^* T_{n+1}(x)$$

故  $p_n^*(x)$  是  $f(x)$  在  $[-1, 1]$  内近似最佳一致逼近多项式。

**例 7.3.6** 设  $f(x) = e^x$ , 在  $[-1, 1]$  内按切比雪夫多项式  $\{T_k(x)\}_0^{\infty}$  展开, 并计算到  $n=3$ 。

**解**

$$a_k^* = \frac{2}{\pi} \int_{-1}^1 \frac{e^x T_k(x)}{\sqrt{1-x^2}} dx = \frac{2}{\pi} \int_0^{\pi} e^{\cos \theta} \cos k\theta d\theta$$

用数值积分计算, 结果见表 7.3.1。

表 7.3.1 积分值  $a_k^*$

$k$	0	1	2	3	4
$a_k^*$	2.532 132	1.130 318	0.271 495	0.044 336 9	0.005 474 24

$$\begin{aligned} p_3^*(x) &= \frac{1}{2}a_0^* + a_1^*T_1(x) + a_2^*T_2(x) + a_3^*T_3(x) = \cdots \\ &= 0.994571 + 0.99730x + 0.5429991x^2 + 0.177347x^3 \end{aligned}$$

## 7.4 最佳平方逼近

### 7.4.1 基本概念

**定义 7.4.1** (连续函数线性空间) 设  $\varphi_1(x), \varphi_2(x), \dots, \varphi_n(x)$  是在  $[a, b]$  内的线性无关的连续函数,  $a_0, a_1, \dots, a_n$  是任意实数, 则

$$\Phi = \{S(x) \text{ 的全体} \mid S(x) = a_0\varphi_0(x) + \cdots + a_n\varphi_n(x), a_i \in R, i = 0, 1, \dots, n\}$$

(7.4.1)

称集合  $\Phi$  为由  $\varphi_1(x), \varphi_2(x), \dots, \varphi_n(x)$  所生成的线性空间, 称  $\varphi_1(x), \varphi_2(x), \dots, \varphi_n(x)$  为生成线性空间  $\Phi$  的一个基底。

**定理 7.4.1** 设  $\varphi_1(x), \varphi_2(x), \dots, \varphi_n(x) \in [a, b]$ , 则  $\varphi_1(x), \varphi_2(x), \dots, \varphi_n(x)$  在  $[a, b]$  内线性无关的充分必要条件是  $\det \mathbf{G}_n \neq 0$ 。

$$\mathbf{G}_n = \mathbf{G}(\varphi_0, \varphi_1, \dots, \varphi_n) = \begin{vmatrix} (\varphi_0, \varphi_0) & (\varphi_0, \varphi_1) & \cdots & (\varphi_0, \varphi_n) \\ (\varphi_1, \varphi_0) & (\varphi_1, \varphi_1) & \cdots & (\varphi_1, \varphi_n) \\ \vdots & \vdots & \ddots & \vdots \\ (\varphi_n, \varphi_0) & (\varphi_n, \varphi_1) & \cdots & (\varphi_n, \varphi_n) \end{vmatrix} \quad (7.4.2)$$

式中,  $(\varphi_i, \varphi_j)$  表示  $\varphi_i(x)$  与  $\varphi_j(y)$  的内积 ( $i, j = 1, 2, \dots, n$ )。

特别地, 若取  $\varphi_k(x) = x^k, k = 0, 1, \dots, n$ , 区间取  $[0, 1]$ ,  $\rho(x) = 1$ ,  $f(x) \in C[0, 1]$ , 则  $\Phi = P_n = \text{span}\{1, x, \dots, x^n\}$  为  $n$  次多项式集合, 且  $P_n(x) = a_0 + a_1x + \cdots + a_nx^n$ 。

此时由于

$$(\varphi_j, \varphi_k) = \int_0^1 x^{j+k} dx = \frac{1}{j+k+1} \quad (j, k = 0, 1, \dots, n) \quad (7.4.3)$$

则  $\mathbf{G}_n$  记为  $\mathbf{H}_n$

$$\mathbf{H}_n = \begin{bmatrix} 1 & 1/2 & \cdots & 1/(n+1) \\ 1/2 & 1/3 & \cdots & 1/(n+2) \\ \vdots & \vdots & \ddots & \vdots \\ 1/(n+1) & 1/(n+2) & \cdots & 1/(2n+1) \end{bmatrix} = (h_{ij})_{(n+1) \times (n+1)} \quad (7.4.4)$$

式中,  $h_{ij} = \frac{1}{i+j-1}$ ,  $\mathbf{H}_n$  称为希尔伯特 (Hilbert) 矩阵。

## 7.4.2 最佳平方逼近函数

定义 7.4.2 (最佳平方逼近函数) 设  $f(x) \in C[a, b]$ , 如果存在  $S^*(x) \in \Phi$  使

$$\int_a^b \rho(x)[f(x) - S^*(x)]^2 dx = \min_{S(x) \in \Phi} \int_a^b \rho(x)[f(x) - S(x)]^2 dx \quad (7.4.5)$$

则称  $S^*(x)$  是  $f(x)$  在集合  $\Phi$  中的最佳平方逼近函数。

若  $\Phi = P_n = \text{span}\{1, x, \dots, x^n\}$ , 则满足定义 7.4.2 的  $S^*(x)$  是  $f(x)$  的  $n$  次最佳平方逼近多项式。

定理 7.4.2 (存在唯一性定理) 设  $f(x) \in C[a, b]$ , 则  $f(x)$  在  $\Phi$  中存在唯一的最佳平方逼近函数  $S^*(x)$ 。

证明 先构造函数  $S^*(x)$ , 求  $S^*(x) \in \Phi$  等价于求

$$I(a_0, a_1, \dots, a_n) = \int_a^b \rho(x) \left[ \sum_{j=0}^n a_j \varphi_j(x) - f(x) \right]^2 dx$$

关于  $a_0, a_1, \dots, a_n$  的极小值。其必要条件是偏导数为 0, 即

$$\frac{\partial I}{\partial a_k} = 2 \int_a^b \rho(x) \left( \sum_{j=0}^n a_j \varphi_j(x) - f(x) \right) \varphi_k(x) dx = 0 \quad (k = 0, 1, \dots, n)$$

于是有

$$\sum_{j=0}^n (\varphi_j(x), \varphi_k(x)) a_j = (f(x), \varphi_k(x)) \quad (k = 0, 1, \dots, n) \quad (7.4.6)$$

这是关于  $a_0, a_1, \dots, a_n$  的线性方程组, 称为法方程。

由于  $\varphi_1(x), \varphi_2(x), \dots, \varphi_n(x)$  线性无关, 方程组有唯一的解:  $a_k = a_k^* (k = 0, 1, \dots, n)$

于是, 有

$$S^*(x) = a_0^* \varphi_0(x) + a_1^* \varphi_1(x) + \dots + a_n^* \varphi_n(x)$$

再证明  $S^*(x)$  即为最佳平方逼近函数, 即  $S^*(x)$  满足式 (7.4.5), 为此只需要考虑:

$$\begin{aligned} D &= \int_a^b \rho(x)[f(x) - S(x)]^2 dx - \int_a^b \rho(x)[f(x) - S^*(x)]^2 dx \\ &= \int_a^b \rho(x)[S(x) - S^*(x)]^2 dx + 2 \int_a^b \rho(x)[S^*(x) - S(x)][f(x) - S^*(x)] dx \end{aligned} \quad (7.4.7)$$

由于  $S^*(x)$  的系数  $a_j^* (j = 0, 1, \dots, n)$  是方程组 (7.4.6) 的解, 因此有

$$\int_a^b \rho(x)[f(x) - S^*(x)] \varphi_k(x) dx = 0 \quad (k = 0, 1, \dots, n)$$

从而式 (7.4.7) 第二个积分为 0, 于是  $D = \int_a^b \rho(x)[S(x) - S^*(x)]^2 dx \geq 0$ , 所以式 (7.4.5) 成立。

这就证明了  $S^*(x)$  是  $f(x)$  在  $\Phi$  中的最佳平方逼近函数。

作为特例, 若取  $\varphi_k(x) = x^k (k = 0, 1, \dots, n)$ , 区间取  $[0, 1]$ ,  $\rho(x) = 1$ ,  $f(x) \in C[0, 1]$ , 在  $\Phi = P_n = \text{span}\{1, x, \dots, x^n\}$  上的最佳平方逼近多项式为

$$P_n^*(x) = a_0^* + a_1^* x + \dots + a_n^* x^n$$

此时由于

$$(\varphi_j, \varphi_k) = \int_0^1 x^{j+k} dx = \frac{1}{j+k+1} \quad (j, k = 0, 1, \dots, n)$$

$$(f, \varphi_k) = \int_0^1 f(x) x^k dx = d_k \quad (k = 0, 1, \dots, n)$$

则法方程 (7.4.6) 的系数矩阵  $G_n$  为式 (7.4.4) 的  $H_n$ 。



若记  $\mathbf{a} = (a_0, a_1, \dots, a_n)^\top$ ,  $\mathbf{d} = (d_0, d_1, \dots, d_n)^\top$ , 则法方程可记为

$$\mathbf{H}_n \mathbf{a} = \mathbf{d} \quad (7.4.8)$$

其解为  $a_k = a_k^* (k=0, 1, \dots, n)$ , 由此可得最佳平方逼近多项式  $P_n^*(x)$ 。

**例 7.4.1** 设  $f(x) = \sqrt{1+x^2}$ , 在  $[0, 1]$  内求  $f(x)$  的最佳平方逼近多项式  $P_1(x) = a_0^* + a_1^* x$ 。

**解**

$$d_0 = \int_0^1 \sqrt{1+x^2} dx = \frac{1}{2} \ln(1+\sqrt{2}) - \frac{\sqrt{2}}{2} \approx 1.147$$

$$d_1 = \int_0^1 \sqrt{1+x^2} x dx = \frac{2\sqrt{2}-1}{3} \approx 0.609$$

法方程为

$$\begin{bmatrix} 1 & 1/2 \\ 1/2 & 1/3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 1.147 \\ 0.609 \end{bmatrix}$$

求得解为  $a_0^* = 0.934$ ,  $a_1^* = 0.426$ , 于是得

$$P_1^*(x) = 0.934 + 0.426x$$

由于式 (7.4.8) 是病态方程, 此时, 可用正交多项式作  $\Phi$  的基的方法求解最佳平方逼近多项式。设  $f(x) \in C[a, b]$ ,  $\Phi = \text{span}\{\varphi_0, \varphi_1, \dots, \varphi_n\}$ ,  $\varphi_0, \varphi_1, \dots, \varphi_n$  是正交函数, 则当  $i \neq j$  时,  $(\varphi_i, \varphi_j) = 0$ , 而  $(\varphi_j, \varphi_j) > 0$ , 于是, 法方程的系数矩阵  $\mathbf{G}_n$  为非奇异对角阵, 方程的解为

$$a_k^* = \frac{(f, \varphi_k)}{(\varphi_k, \varphi_k)} \quad (k=0, 1, \dots, n) \quad (7.4.9)$$

于是,  $f(x)$  在  $\Phi$  中的最佳平方逼近函数为

$$\varphi^*(x) = \sum_{k=0}^n \frac{(f, \varphi_k)}{\|\varphi_k\|_2^2} \varphi_k(x) \quad (7.4.10)$$

式 (7.4.10) 称为傅里叶 (Fourier) 展开,  $a_k^*$  为广义傅里叶系数。

特别地, 设  $[a, b] = [-1, 1]$ ,  $\rho(x) = 1$ , 此时, 正交多项式为勒让德多项式。

取  $\Phi = \text{span}\{P_0(x), P_1(x), \dots, P_n(x)\}$ , 可得  $f(x) \in C[-1, 1]$  的最佳平方逼近多项式为

$$S_n^*(x) = \sum_{k=0}^n a_k P_k(x) \quad (7.4.11)$$

式中

$$a_k = \frac{(f, P_k)}{\|P_k\|_2^2} = \frac{2k+1}{2} \int_{-1}^1 f(x) P_k(x) dx \quad (7.4.12)$$

这样得到的  $S_n^*(x)$  与直接由  $\{1, x, \dots, x^n\}$  为基得到的  $P_n^*(x)$  是一致的, 但此时不用解病态方程组 (7.4.8)。

**例 7.4.2** 用勒让德正交多项式求  $f(x) = e^x$  在  $[-1, 1]$  内的最佳平方逼近多项式 (取  $n=1, 3$ )

**解** 先计算  $(f, P_0) = \int_{-1}^1 e^x dx = e - e^{-1} \approx 2.3504$

$$(f, P_1) = \int_{-1}^1 e^x x dx = 2e^{-1} \approx 0.7358$$

$$(f, P_2) = \int_{-1}^1 \left( \frac{3}{2}x^2 - \frac{1}{2} \right) e^x dx = e - 7e^{-1} \approx 0.1431$$

$$(f, P_3) = \int_{-1}^1 \left( \frac{5}{2}x^3 - \frac{3}{2}x \right) e^x dx = 37e^{-1} - 5e \approx 0.02013$$

于是, 由式 (7.4.12) 可求得

$$a_0^* = 1.1752, \quad a_1^* = 1.1036, \quad a_2^* = 0.3578, \quad a_3^* = 0.07046$$

所以由式 (7.4.11) 得:

$$S_1^*(x) = 1.1752 + 1.1036x$$

$$S_3^*(x) = 0.9963 + 0.9979x + 0.5367x^2 + 0.1761x^3$$

如果所给区间不是  $[-1, 1]$ , 而是一般的有限区间  $[a, b]$ , 那么可以通过变量转换  $x = \frac{a+b}{2} + \frac{b-a}{2}t$ ,

将它转化为区间  $-1 \leq t \leq 1$  内的情况来处理。

**例 7.4.3** 求  $f(x) = \sqrt{x}$  在  $[0, 1]$  内的一次最佳平方逼近多项式。

**解** 令  $x = \frac{1}{2}(1+t)$

则 
$$f(x) = \frac{\sqrt{1+t}}{\sqrt{2}} = g(t)$$

下面求  $g(t)$  在区间  $[-1, 1]$  内的一次最佳平方逼近多项式  $g_1(t)$ 。

$$\text{由 } a_0^* = \frac{1}{2}(g, P_0) = \frac{1}{2} \int_{-1}^1 \frac{1}{\sqrt{2}} \sqrt{1+t} dt = \frac{2}{3}$$

$$a_1^* = \frac{3}{2}(g, P_1) = \frac{3}{2} \int_{-1}^1 \frac{t}{\sqrt{2}} \sqrt{1+t} dt = \frac{6}{15}$$

$$\text{可知 } g_1(t) = \frac{2}{3}P_0 + \frac{6}{15}P_1 = \frac{2}{3} + \frac{6}{15}t \quad (-1 \leq t \leq 1)$$

再把  $t = 2x - 1$  代入  $g_1(t)$ , 就得到  $f(x) = \sqrt{x}$  在  $[0, 1]$  内的一次最佳平方逼近多项式

$$S^*(x) = \frac{2}{3} + \frac{6}{15}(2x-1) = \frac{4}{15} + \frac{12}{15}x$$

## 7.5 离散数据的曲线拟合

### 7.5.1 曲线拟合问题

在科学实验和统计研究中, 往往需要从一组测量得到的数据  $(x_i, y_i) (i = 0, 1, 2, \dots, n)$  中估计函数关系  $y = f(x)$  的近似函数  $p(x)$ 。插值方法就是处理这类问题的一种方法。但是, 它有明显的缺点。首先, 因为由观测得到的实验数据不可避免地带有误差, 甚至是较大的误差, 此时要求近似函数  $p(x)$  过全部已知点, 相当于保留全部的误差。另外, 由于实验数据往往很多, 如果仍然采用多项式插值, 必然得到次数较高的多项式, 这样, 不但计算复杂, 而且  $p(x)$  的收敛性和稳定性都很难得到保证, 会出现龙格现象, 逼近的效果不佳。

为了克服上述缺点, 常常采用曲线拟合的方法来进行数据处理。所谓曲线拟合就是从数据集  $(x_i, y_i) (i = 0, 1, 2, \dots, n)$  中找出总体规律性, 并构造一条能较好反映这种规律的曲线  $p(x)$ , 此时, 并不要求曲线  $p(x)$  过每个数据点  $(x_i, y_i) (i = 0, 1, 2, \dots, n)$ , 但要求曲线  $p(x)$  能尽可能地靠近所有数据点, 即所有误差  $\delta_i = p(x_i) - y_i (i = 0, 1, 2, \dots, n)$  都按某种标准达到最小。一般采用以下 3 种标准来度量误差的大小。

$$\textcircled{1} \quad \|\delta\|_1 = \sum_{i=0}^n |\delta_i|$$

$$\textcircled{2} \quad \|\delta\|_2^2 = \sum_{i=0}^n \delta_i^2$$

$$\textcircled{3} \quad \|\delta\|_\infty = \max_{0 \leq i \leq n} |\delta_i|$$

由于 2-范数中没有绝对值, 计算过程比较方便, 因此通常采用 2-范数的平方作为总体误差的度量标准。

## 7.5.2 多项式拟合

对于给定的一组数据  $(x_i, y_i) (i=0, 1, 2, \dots, n)$ , 在函数类  $\Phi = \{\varphi_0(x), \varphi_1(x), \dots, \varphi_m(x)\}$  中寻求一个函数  $p(x)$ , 使误差的 2-范数平方

$$\|\delta\|_2^2 = \sum_{i=0}^n \delta_i^2 = \sum_{i=0}^n (p(x_i) - y_i)^2 \quad (7.5.1)$$

达到最小, 设  $\varphi_0(x), \varphi_1(x), \dots, \varphi_m(x)$  是  $\Phi$  的一组线性无关的基函数,  $p(x)$  为  $\{\varphi_i(x)\} (i=0, 1, \dots, m)$  的线性组合, 即

$$p(x) = a_0 \varphi_0(x) + a_1 \varphi_1(x) + \dots + a_m \varphi_m(x) \quad (m < n) \quad (7.5.2)$$

将式 (7.5.2) 代入式 (7.5.1) 中, 使误差的 2-范数平方  $\|\delta\|_2^2$  取最小值问题转化为求下列多元函数

$$F(a_0, a_1, \dots, a_m) = \sum_{i=0}^n \left( \sum_{j=0}^m a_j \varphi_j(x_i) - y_i \right)^2$$

的极小点  $(a_0^*, a_1^*, \dots, a_m^*)$ , 即令

$$\frac{\partial F}{\partial a_k} = 0 \quad (k=0, 1, 2, \dots, m)$$

由此得

$$\sum_{i=0}^n \left( \sum_{j=0}^m a_j \varphi_j(x_i) - y_i \right) \varphi_k(x_i) = 0 \quad (k=0, 1, 2, \dots, m) \quad (7.5.3)$$

若用离散意义下函数的内积符号

$$(\varphi_j, \varphi_k) = \sum_{i=0}^n \varphi_j(x_i) \varphi_k(x_i), \quad (f, \varphi_k) = \sum_{i=0}^n y_i \varphi_k(x_i) = d_k$$

表示, 则式 (7.5.3) 可写为

$$\sum_{j=0}^m (\varphi_k, \varphi_j) a_j = d_k \quad (k=0, 1, 2, \dots, m)$$

其矩阵的形式为

$$\begin{pmatrix} (\varphi_0, \varphi_0) & (\varphi_0, \varphi_1) & \cdots & (\varphi_0, \varphi_m) \\ (\varphi_1, \varphi_0) & (\varphi_1, \varphi_1) & \cdots & (\varphi_1, \varphi_m) \\ \vdots & \vdots & \ddots & \vdots \\ (\varphi_m, \varphi_0) & (\varphi_m, \varphi_1) & \cdots & (\varphi_m, \varphi_m) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_m \end{pmatrix} \quad (7.5.4)$$

式 (7.5.4) 是关于系数  $a_j (j=0, 1, 2, \dots, m)$  的线性方程组, 也称为法方程。由于  $\varphi_0, \varphi_1, \dots, \varphi_m$  线性

无关, 由定理 7.4.1 可知, 式 (7.5.4) 的系数矩阵的行列式不为 0, 因此方程组 (7.5.4) 有唯一的解  $(a_0^*, a_1^*, a_2^*, \dots, a_m^*)$ 。

当函数类  $\Phi = \{\varphi_0, \varphi_1, \dots, \varphi_m\} = \{1, x, x^2, \dots, x^m\}$  时,

$$p(x) = a_0 + a_1x + \dots + a_mx^m$$

为  $m$  次多项式。相应地, 曲线拟合成为多项式拟合, 法方程 (7.5.4) 可写为

$$\begin{pmatrix} \sum 1 & \sum x_i & \cdots & \sum x_i^m \\ \sum x_i & \sum x_i^2 & \cdots & \sum x_i^{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum x_i^m & \sum x_i^{m+1} & \cdots & \sum x_i^{2m} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum x_i y_i \\ \vdots \\ \sum x_i^m y_i \end{pmatrix} \quad (7.5.5)$$

不过, 实际计算和理论分析表明, 当  $n$  较大时, 方程组 (7.5.5) 为“病态”方程组。

**例 7.5.1** 已知实验数据如下:

$i$	0	1	2	3
$x_i$	2	4	6	8
$y_i$	2	11	28	40

求拟合曲线  $y = p(x)$ 。

**解** (1) 在坐标平面上描出点  $(x_i, y_i)$  ( $i = 0, 1, 2, 3$ ), 如图 7.5.1 所示。

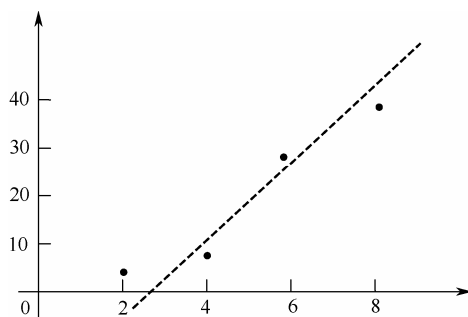


图 7.5.1 在平面上描点

(2) 根据各点的分布情况, 选用线性函数

$$p(x) = a_0 + a_1x$$

作拟合曲线, 故取  $\varphi_0(x) = 1, \varphi_1(x) = x$ 。

(3) 建立法方程, 因为

$$(\varphi_0, \varphi_0) = \sum 1 = 4, \quad (\varphi_0, \varphi_1) = (\varphi_1, \varphi_0) = \sum x_i = 20$$

$$(\varphi_1, \varphi_1) = \sum x_i^2 = 120, \quad (f, \varphi_0) = d_0 = \sum y_i = 81$$

$$(f, \varphi_1) = d_1 = \sum x_i y_i = 536$$

所以法方程为

$$\begin{pmatrix} 4 & 20 \\ 20 & 120 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} 81 \\ 536 \end{pmatrix}$$

解得

$$a_0 = -12.5, \quad a_1 = 6.55$$

于是

$$p(x) = -12.5 + 6.55x$$

例 7.5.2 已知函数表如下:

$x$	-2	-1	0	1	2
$y$	0	1	2	1	0

试用二次多项式  $p(x) = a_0 + a_1x + a_2x^2$  拟合这组数据。

解 (1) 记点  $-2, -1, 0, 1, 2$  分别为  $x_0, x_1, x_2, x_3, x_4$ ,  $\varphi_0(x) = 1$ ,  $\varphi_1(x) = x$ ,  $\varphi_2(x) = x^2$ 。

(2) 法方程的系数矩阵为

$$G = \begin{pmatrix} (1,1) & (1,x) & (1,x^2) \\ (x,1) & (x,x) & (x,x^2) \\ (x^2,1) & (x^2,x) & (x^2,x^2) \end{pmatrix} = \begin{pmatrix} 5 & 0 & 10 \\ 0 & 10 & 0 \\ 10 & 0 & 34 \end{pmatrix}$$

法方程的右端项为

$$((y,1), (y,x), (y,x^2))^T = (4, 0, 2)^T$$

于是解法方程

$$\begin{pmatrix} 5 & 0 & 10 \\ 0 & 10 & 0 \\ 10 & 0 & 34 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 4 \\ 0 \\ 2 \end{pmatrix}$$

得

$$a_0 = \frac{58}{35}, \quad a_1 = 0, \quad a_2 = -\frac{3}{7}$$

这样, 求得拟合多项式为

$$p(x) = \frac{58}{35} - \frac{3}{7}x^2$$

从以上两个例子可以看出, 求拟合多项式的步骤如下。

① 由给定的数据在坐标纸上进行描点, 根据点的分布, 确定拟合多项式的次数。

② 求解法方程 (7.5.5), 求出系数  $a_i^*$  ( $i = 0, 1, 2, \dots, n$ )。

③ 写出拟合多项式  $S(x) = \sum_{i=0}^n a_i x^i$ 。

### 7.5.3 正交多项式拟合

由于当  $n$  较大时, 用多项式拟合时所求的法方程 (7.5.5) 一般为“病态”方程组, 所以为了避免求解法方程 (7.5.5), 可以用 7.2 节的相关知识, 进行正交多项式曲线拟合。

例 7.5.3 构造关于离散点集  $\{-2, -1, 0, 1, 2\}$ , 权系数为  $\{1, 1, 1, 1, 1\}$ , 首项系数为 1 的正交多项式族  $p_0(x), p_1(x), p_2(x)$ 。

解 记  $-2, -1, 0, 1, 2$  分别为  $x_0, x_1, x_2, x_3, x_4$ , 在离散节点  $\{x_i\}$  ( $i = 0, 1, \dots, 4$ ) 上的内积为

$$(f, g) = \sum_{i=0}^4 f(x_i) \cdot g(x_i)$$

利用式 (7.2.5) 和式 (7.2.6) 得

$$\begin{aligned} p_0(x) &= 1 \\ \alpha_0 &= \frac{(xp_0, p_0)}{(p_0, p_0)} = \frac{0}{5} = 0 \\ p_1(x) &= (x - \alpha_0)p_0(x) = x \\ \alpha_1 &= \frac{(xp_1, p_1)}{(p_1, p_1)} = \frac{0}{10} = 0 \\ \beta_0 &= \frac{(p_1, p_1)}{(p_0, p_0)} = \frac{10}{5} = 2 \\ p_2(x) &= (x - \alpha_1)p_1 - \beta_0 p_0 = x^2 - 2 \end{aligned}$$

一般地, 当取关于点集  $\{x_i\} (i=0, 1, 2, \dots, m)$ , 带权  $\{\omega_i\} (i=0, 1, 2, \dots, m)$  正交多项式族  $p_0(x), p_1(x), \dots, p_n(x)$  作为拟合多项式的函数类  $\Phi$  时, 由正交多项式的性质, 法方程可以简化为

$$(p_k, p_k)a_k = (f, p_k) \quad (k=0, 1, 2, \dots, n) \quad (7.5.6)$$

求解得

$$a_k = \frac{(f, p_k)}{(p_k, p_k)} \quad (k=0, 1, 2, \dots, n) \quad (7.5.7)$$

则  $n$  次拟合多项式为

$$p(x) = \sum_{k=0}^n a_k p_k(x) \quad (7.5.8)$$

**例 7.5.4** 用正交多项式族, 求例 7.5.2 的拟合多项式。

**解** (1) 建立关于节点  $\{-2, -1, 0, 1, 2\}$  的正交多项式族, 由例 7.5.3 可知

$$p_0(x) = 1, p_1(x) = x, p_2(x) = x^2 - 2$$

(2) 利用式 (7.5.8), 得二次拟合多项式为

$$\begin{aligned} p(x) &= \frac{(y, p_0)}{(p_0, p_0)} p_0(x) + \frac{(y, p_1)}{(p_1, p_1)} p_1(x) + \frac{(y, p_2)}{(p_2, p_2)} p_2(x) \\ &= \frac{4}{5} \times 1 + \frac{0}{10} \times x + \frac{-6}{14} (x^2 - 2) = \frac{58}{35} - \frac{3}{7} x^2 \end{aligned}$$

计算结果与例 7.5.2 相同, 但此例不需要解法方程 (7.5.5), 而只需要计算内积, 避免了出现病态方程组的可能, 并且当逼近次数增加一次时, 只需要在原有的拟合多项式中增加一项即可, 节省了大量计算。

## 本章小结

本章介绍了复杂函数的最佳一致逼近、最佳平方逼近和曲线拟合问题。前两种是复杂函数在连续区间上的近似表示形式, 后一种是复杂函数在离散节点上的近似表示形式, 它们都要求复杂函数与近似函数在整体上某种误差度量最小。正交多项式是求近似函数的有利工具。

用一个函数序列  $p_n(x)$  逼近函数  $f(x)$ , 一致逼近要求函数序列  $p_n(x)$  满足  $\lim_{n \rightarrow \infty} \|p_n(x) - f(x)\|_{\infty} = 0$ , 平方逼近要求函数序列  $\{p_n(x)\}$  满足  $\lim_{n \rightarrow \infty} \|p_n(x) - f(x)\|_2^2 = 0$ , 由于在数值上求  $f(x)$  的一致逼近和平方逼近很难, 因此在实际应用中, 感兴趣的是最佳一致逼近和最佳

平方逼近。离散曲线拟合要求以离散节点的平方和  $\|p-f\|_2^2 = \sum_{i=0}^n (p(x_i) - f(x_i))^2$  作为度量逼近函数  $p(x)$  和被逼近函数  $f(x)$  的逼近程度。

## 习题 7

7.1 证明函数  $1, x, \dots, x^n$  线性无关。

7.2 计算函数  $f(x) = (x-1)^3$  关于  $C[0,1]$  的  $\|f(x)\|_\infty$ 、 $\|f(x)\|_1$  和  $\|f(x)\|_2$ 。

7.3 对权函数  $\rho(x) = 1+x^2$ ，区间取  $[-1,1]$ ，试求首项系数为 1 的正交多项式  $\varphi_n(x)$  ( $n=0,1,2,3$ )。

7.4 求  $f(x) = e^x$  在区间  $[0,1]$  内的最佳一次逼近多项式。

7.5 求  $f(x) = x^4 + 3x^3 - 1$  在区间  $[0,1]$  内的三次最佳一致逼近多项式。

7.6 求函数  $f(x) = \frac{1}{x}$  在区间  $[1,3]$  内对于  $\Phi = \text{span}\{1, x\}$  的最佳平方逼近多项式。

7.7 已知点序  $\{x_i\}_{i=0}^m = \{-2, -1, 0, 1, 2\}$  和权数  $\{w_i\}_{i=0}^m = \{0.5, 1, 1, 1, 1.5\}$ ，试构造对应的正交多项式  $p_0(x), p_1(x), p_2(x)$ 。

7.8 用最小二乘法求一个形如  $y = a + bx^2$  的经验公式，使它拟合下列数据，并计算均方误差。

$x_i$	19	25	31	38	44
$y_i$	19.0	32.3	49.0	73.3	97.8

7.9 用下列数据构造一次多项式  $y = ax + b$ 。

$x_i$	-2	-1	0	1	2
$y_i$	0	0.2	0.5	0.8	1

7.10 用最小二乘法解下列超定方程

$$\begin{cases} 2x + 4y = 11 \\ 3x - 5y = 3 \\ x + 2y = 6 \\ 2x + y = 7 \end{cases}$$

## 第8章 数值积分与数值微分



### 学习要点

数值积分是将被积函数值的线性组合作为定积分的近似值,数值微分是将微分离散化为差分进行计算,本章内容有:

(1) 代数精度的概念及其求法。

(2) 数值求积公式,包括等距节点的牛顿-柯特斯及其复化求积公式、收敛速度较快的龙贝格求积公式和精度最高的高斯求积公式。

(3) 数值微分公式,包括中点方法、插值型求导公式。



### 教学建议

本章首先要求学生了解数值求积公式的思想和意义,理解代数精度的概念,熟练掌握基于插值的牛顿-柯特斯及其复化求积公式,以及基于外推原理的龙贝格求积公式。高斯求积公式具有最高的代数精度,但求高斯点的过程复杂,可作为选学内容。求数值微分的困难在于步长的选取。建议学时为4~6学时。

## 8.1 引言

### 8.1.1 数值求积的必要性

在工程技术和科学研究中,经常需要计算定积分。许多数学问题,如微分方程和积分方程的求解等也都需要计算定积分,所以,在区间 $[a,b]$ 内求定积分

$$I(f) = \int_a^b f(x) dx \quad (8.1.1)$$

是一个比较传统但应用广泛的问题。从理论上讲,要计算连续函数 $f(x)$ 在 $[a,b]$ 内的定积分,只要找到 $f(x)$ 的原函数 $F(x)$  ( $F'(x) = f(x)$ ),则由牛顿-莱布尼兹(Newton-Leibniz)公式

$$\int_a^b f(x) dx = F(b) - F(a) \quad (8.1.2)$$

很容易求出定积分式(8.1.1)。但在实际中,这一重要定理的应用却遇到了很大困难,因为在实际中被积函数 $f(x)$ 的表现形式比较复杂,通常有下列三种形式。

① 被积函数 $f(x)$ 的原函数不能用初等函数的有限形式表示,如 $\int_0^1 \frac{\sin x}{x} dx, \int_0^1 e^{-x^2} dx$ 等,所以不能应用式(8.1.2)进行定积分的计算。

② 被积函数 $f(x)$ 是用函数表的形式或图形的形式给出的,没有解析表达式,因此也就没有应用式(8.1.2)的可能性。



③ 虽然被积函数  $f(x)$  的原函数能用初等函数表示, 但由于其表达式过于复杂, 计算定积分的值会非常困难, 因此, 很难用式 (8.1.2) 计算定积分。

对于以上三种情况, 都必须通过近似的方法计算, 所以, 有必要研究定积分的数值计算问题。

## 8.1.2 数值积分的基本思想

定积分式 (8.1.1) 的几何意义是由曲线  $y = f(x)$ , 直线  $x = a, x = b$  与  $x$  轴所围成的曲边梯形的面积, 因此, 不论被积函数以什么形式给出, 只要近似计算出相应曲边梯形的面积, 就可以求出定积分式 (8.1.1) 的近似值, 这就是数值求积的基本思想。

矩形法: 用分点  $a = x_0 < x_1 < \cdots < x_n = b$ , 将区间  $[a, b]$  进行划分, 记

$$\Delta x_i = x_{i+1} - x_i, f(x_i) = f_i$$

若取  $f_i$  为  $[x_i, x_{i+1}]$  内矩形的高, 则得左矩形公式

$$I(f) = \int_a^b f(x) dx \approx \Delta x_0 f_0 + \Delta x_1 f_1 + \cdots + \Delta x_{n-1} f_{n-1} + 0 f_n$$

若取  $f_{i+1}$  为  $[x_i, x_{i+1}]$  内矩形的高, 则得右矩形公式

$$I(f) = \int_a^b f(x) dx \approx 0 f_0 + \Delta x_0 f_1 + \Delta x_1 f_2 + \cdots + \Delta x_{n-1} f_n$$

取上两式的平均值, 得梯形公式

$$I(f) = \int_a^b f(x) dx \approx \frac{\Delta x_0}{2} f_0 + \frac{\Delta x_0 + \Delta x_1}{2} f_1 + \cdots + \frac{\Delta x_{n-2} + \Delta x_{n-1}}{2} f_{n-1} + \frac{\Delta x_{n-1}}{2} f_n$$

上述三个公式的共同特点是将定积分的计算转换为计算各点函数值的线性组合, 只是各函数值的系数不同而已。一般地, 在  $[a, b]$  内适当选取  $n+1$  个节点  $x_i (i=0, 1, \cdots, n)$ , 然后用  $f(x_i)$  的线性组合作为定积分的近似值。所以, 数值求积公式的一般形式为

$$I(f) = \int_a^b f(x) dx \approx \sum_{i=0}^n A_i f(x_i) = I_n(f) \quad (8.1.3)$$

在式 (8.1.3) 中,  $x_i$  称为求积节点,  $A_i$  称为求积系数,  $A_i$  的值仅与节点  $x_i$  的选取有关, 而不依赖于被积函数  $f(x)$ , 因此式 (8.1.3) 具有通用性, 其特点是直接用一些离散节点上的函数值  $f(x_i)$  的线性组合来计算定积分的近似值, 从而将定积分的计算归纳为函数值的计算, 这就避开了式 (8.1.2) 中需要原函数的困难, 并为用计算机编程求积分的近似值提供了可行性。

## 8.1.3 代数精度

数值求积公式 (8.1.3) 是近似求积方法, 为了保证精度, 自然希望所提供的求积公式对于“尽可能多”的函数是准确的, 这就提出了代数精度的概念。

**定义 8.1.1 (代数精度)** 若数值求积公式 (8.1.3) 对被积函数  $f(x) = 1, x, \cdots, x^m$  都能精确成立, 而对被积函数  $f(x) = x^{m+1}$  不能精确成立, 则称求积公式 (8.1.3) 具有  $m$  次代数精度。

一般地, 要使式 (8.1.3) 具有  $n$  次代数精度, 只要令它对于  $f(x) = 1, x, x^2, \cdots, x^n$  都精确成立, 也就是对给定  $n+1$  个互异节点  $x_i (i=0, 1, \cdots, n)$ , 相应的求积系数  $A_i$  满足

$$\begin{cases} A_0 + A_1 + \cdots + A_n = b - a \\ A_0 x_0 + A_1 x_1 + \cdots + A_n x_n = \frac{b^2 - a^2}{2} \\ A_0 x_0^n + A_1 x_1^n + \cdots + A_n x_n^n = \frac{b^{n+1} - a^{n+1}}{n+1} \end{cases} \quad (8.1.4)$$

线性方程组 (8.1.4) 的系数行列式是范德蒙行列式, 当  $x_i (i=0, 1, \cdots, n)$  互异时, 其值非零 (参见例 2.3.3), 利用克莱姆法则可以唯一地求出  $A_i (i=0, 1, \cdots, n)$ , 进而可以构造出求积公式 (8.1.3), 于是, 有如下结论。

**定理 8.1.1** 对于任意给定  $n+1$  个互异节点  $x_i (i=0, 1, \cdots, n)$ , 总存在相应系数  $A_i (i=0, 1, \cdots, n)$ , 使得式 (8.1.3) 至少具有  $n$  次代数精度。

**例 8.1.1** 证明求积公式

$$\int_{-1}^1 f(x) dx \approx \frac{1}{9} [5f(\sqrt{0.6}) + 8f(0) + 5f(-\sqrt{0.6})]$$

对于次数不高于 5 的多项式精确成立。

**证明** 由于求积公式中系数与节点全部给定, 可直接将  $f(x) = 1, x, x^2, x^3, x^4, x^5$  代入验证。因为

$$2 = \int_{-1}^1 1 dx = \frac{1}{9} [5 \times 1 + 8 \times 1 + 5 \times 1] = 2$$

$$0 = \int_{-1}^1 x dx = \frac{1}{9} [5 \times \sqrt{0.6} + 8 \times 0 + 5 \times (-\sqrt{0.6})] = 0$$

$$\frac{2}{3} = \int_{-1}^1 x^2 dx = \frac{1}{9} [5 \times (\sqrt{0.6})^2 + 8 \times 0^2 + 5 \times (-\sqrt{0.6})^2] = \frac{2}{3}$$

$$0 = \int_{-1}^1 x^3 dx = \frac{1}{9} [5 \times (\sqrt{0.6})^3 + 8 \times 0^3 + 5 \times (-\sqrt{0.6})^3] = 0$$

$$\frac{2}{5} = \int_{-1}^1 x^4 dx = \frac{1}{9} [5 \times (\sqrt{0.6})^4 + 8 \times 0^4 + 5 \times (-\sqrt{0.6})^4] = \frac{2}{5}$$

$$0 = \int_{-1}^1 x^5 dx = \frac{1}{9} [5 \times (\sqrt{0.6})^5 + 8 \times 0^5 + 5 \times (-\sqrt{0.6})^5] = 0$$

所以求积公式  $\int_{-1}^1 f(x) dx \approx \frac{1}{9} [5f(\sqrt{0.6}) + 8f(0) + 5f(-\sqrt{0.6})]$  对于次数不高于 5 次的多项式精确成立。

**例 8.1.2** 设有求积公式

$$\int_{-1}^1 f(x) dx = A_0 f(-1) + A_1 f(0) + A_2 f(1)$$

试确定系数  $A_0, A_1, A_2$ , 使上述公式的代数精度尽可能高, 并指出该求积公式所具有的代数精度。

**解** 令求积公式依次对  $f(x) = 1, x, x^2$  都精确成立, 即系数  $A_0, A_1, A_2$  满足

$$\begin{cases} A_0 + A_1 + A_2 = 2 \\ -A_0 + A_2 = 0 \\ A_0 + A_2 = \frac{2}{3} \end{cases}$$

解得

$$\begin{cases} A_0 = \frac{1}{3} \\ A_1 = \frac{4}{3} \\ A_2 = \frac{1}{3} \end{cases}$$

因此, 该求积公式为

$$\int_{-1}^1 f(x) dx = \frac{1}{3} f(-1) + \frac{4}{3} f(0) + \frac{1}{3} f(1)$$

不难验证, 该求积公式对于  $f(x) = x^3$  也精确成立, 但对于  $f(x) = x^4$ , 求积公式不能精确成立, 因此, 该求积公式具有 3 次代数精度。

### 8.1.4 插值型求积公式

用求线性方程组 (8.1.4) 的方法构造求积公式 (8.1.3), 计算量非常大, 现在, 用插值多项式来构造数值求积公式 (8.1.3)。

设  $f(x)$  在互异节点  $x_i$  ( $i = 0, 1, 2, \dots, n$ ) 处的函数值为  $f(x_i)$  ( $i = 0, 1, 2, \dots, n$ ), 则可构造拉格朗日插值多项式

$$L_n(x) = \sum_{i=0}^n l_i(x) f(x_i) = \sum_{i=0}^n \left( \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j} \right) f(x_i)$$

由于代数插值多项式  $L_n(x)$  的原函数容易求出, 可取

$$\int_a^b f(x) dx \approx \int_a^b L_n(x) dx = \int_a^b \sum_{i=0}^n l_i(x) f(x_i) dx = \sum_{i=0}^n \left( \int_a^b l_i(x) dx \right) f(x_i)$$

于是得数值求积公式

$$\int_a^b f(x) dx \approx \sum_{i=0}^n A_i f(x_i) \quad (8.1.5)$$

式中

$$A_i = \int_a^b l_i(x) dx = \int_a^b \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j} dx \quad (8.1.6)$$

由式 (8.1.6) 确定的求积公式 (8.1.5) 称为插值型求积公式, 其余项为

$$R[f] = \int_a^b \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{j=0}^n (x - x_j) dx \quad (8.1.7)$$

当被积函数  $f(x)$  取次数不超过  $n$  次的多项式时, 因为  $f^{(n+1)}(x) = 0$ , 所以余项  $R[f] \equiv 0$ , 这说明求积公式 (8.1.5) 对一切次数不超过  $n$  次的多项式都精确成立, 所以含有  $n+1$  个互异节点  $x_i$  ( $i = 0, 1, 2, \dots, n$ ) 的插值型求积公式至少具有  $n$  次代数精度。

反之, 如果求积公式 (8.1.5) 至少有  $n$  次代数精度, 则它对于  $n$  次插值基函数  $l_i(x)$  也是准确成立的, 即有

$$\int_a^b l_i(x) dx = \sum_{j=0}^n A_j l_i(x_j) = A_i$$

因而式 (8.1.6) 成立, 这说明至少具有  $n$  次代数精度的求积公式必为插值型的。

综上所述, 有如下结论。

**定理 8.1.2** 求积公式 (8.1.3) 为插值型求积公式的充分必要条件是它至少具有  $n$  次代数精度。

**例 8.1.3** 已知  $x_0 = \frac{1}{4}$ ,  $x_1 = \frac{1}{2}$ ,  $x_2 = \frac{3}{4}$ , 要求:

(1) 推导以这 3 个点作为求积节点在  $[0,1]$  内的插值型求积公式。

(2) 指明求积公式所具有的代数精度。

(3) 用所求公式计算  $\int_0^1 x^3 dx$ 。

**解** (1) 过这 3 个点的插值多项式

$$p_2(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f(x_0) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f(x_1) + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f(x_2)$$

故

$$\int_0^1 f(x) dx \approx \int_0^1 p_2(x) dx = \sum_{k=0}^2 A_k f(x_k)$$

式中

$$\begin{aligned} A_0 &= \int_0^1 \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} dx = \int_0^1 \frac{\left(x-\frac{1}{2}\right)\left(x-\frac{3}{4}\right)}{\left(\frac{1}{4}-\frac{1}{2}\right)\left(\frac{1}{4}-\frac{3}{4}\right)} dx = \frac{2}{3} \\ A_1 &= \int_0^1 \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} dx = \int_0^1 \frac{\left(x-\frac{1}{4}\right)\left(x-\frac{3}{4}\right)}{\left(\frac{1}{2}-\frac{1}{4}\right)\left(\frac{1}{2}-\frac{3}{4}\right)} dx = -\frac{1}{3} \\ A_2 &= \int_0^1 \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} dx = \int_0^1 \frac{\left(x-\frac{1}{4}\right)\left(x-\frac{1}{2}\right)}{\left(\frac{3}{4}-\frac{1}{4}\right)\left(\frac{3}{4}-\frac{1}{2}\right)} dx = \frac{2}{3} \end{aligned}$$

故所求的插值型求积公式为

$$\int_0^1 f(x) dx \approx \frac{1}{3} \left[ 2f\left(\frac{1}{4}\right) - f\left(\frac{1}{2}\right) + 2f\left(\frac{3}{4}\right) \right]$$

(2) 上述求积公式是由对二次插值函数积分而来的, 故至少具有 2 次代数精度。再将  $f(x) = x^3, x^4$  分别代入上述求积公式, 有

$$\begin{aligned} \frac{1}{4} &= \int_0^1 x^3 dx = \frac{1}{3} \left[ 2\left(\frac{1}{4}\right)^3 - \left(\frac{1}{2}\right)^3 + 2\left(\frac{3}{4}\right)^3 \right] = \frac{1}{4} \\ \frac{1}{5} &= \int_0^1 x^4 dx \neq \frac{1}{3} \left[ 2\left(\frac{1}{4}\right)^4 - \left(\frac{1}{2}\right)^4 + 2\left(\frac{3}{4}\right)^4 \right] \end{aligned}$$

故上述求积公式具有 3 次代数精度。

$$(3) \int_0^1 x^3 dx = \frac{1}{3} \left[ 2\left(\frac{1}{4}\right)^3 - \left(\frac{1}{2}\right)^3 + 2\left(\frac{3}{4}\right)^3 \right] = \frac{1}{4}$$

由于该求积公式具有 3 次代数精度, 从而  $\frac{1}{4}$  为  $\int_0^1 x^3 dx$  的精确值。

## 8.2 牛顿-柯特斯求积公式

### 8.2.1 牛顿-柯特斯公式的导出

设将积分区间  $[a, b]$   $n$  等分, 记步长  $h = \frac{b-a}{n}$ , 求积节点  $x_i$  为等距节点  $x_i = a + ih$  ( $i = 0, 1, 2, \dots, n$ ), 且对应函数值  $f(x_i)$  已知, 则以这些等距节点为插值节点所导出的插值型求积公式就称为牛顿-柯特斯 (Newton-Cotes) 求积公式, 简称 N-C 公式。

为简化计算, 在等距节点下, 将式 (8.1.5) 改写为

$$I(f) = \int_a^b f(x) dx \approx (b-a) \sum_{i=0}^n C_i^{(n)} f(x_i) \quad (8.2.1)$$

比较式 (8.2.1) 与式 (8.1.5), 得

$$C_i^{(n)} = \frac{A_i}{b-a} = \frac{1}{b-a} \int_a^b \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x-x_j}{x_i-x_j} dx$$

为进一步简化计算, 作变换  $x = a + th$ , 则有

$$dx = h dt, \quad x - x_j = (t-j)h, \quad x_i - x_j = (i-j)h$$

从而

$$\begin{aligned} C_i^{(n)} &= \frac{1}{b-a} \int_a^b \frac{(x-x_0)(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)} dx \\ &= \frac{h}{b-a} \int_0^n \frac{t(t-1)\cdots(t-i+1)(t-i-1)\cdots(t-n)h^n}{i(i-1)\cdots 2 \times 1 \times (-1)(-2)\cdots(-(n-i))h^n} dt \\ &= \frac{(-1)^{n-i}}{ni!(n-i)!} \int_0^n t(t-1)\cdots(t-i+1)(t-i-1)\cdots(t-n) dt \end{aligned}$$

即

$$C_i^{(n)} = \frac{(-1)^{n-i}}{ni!(n-i)!} \int_0^n \prod_{\substack{j=0 \\ j \neq i}}^n (t-j) dt \quad (8.2.2)$$

满足等距节点条件下的插值型求积公式 (8.2.1) 称为  $n$  阶牛顿-柯特斯求积公式。式 (8.2.2) 称为柯特斯系数。从式 (8.2.2) 可以看出, 柯特斯系数  $C_i^{(n)}$  只依赖于被积区间  $[a, b]$  的等分数  $n$ , 它与积分区间  $[a, b]$  和被积函数  $f(x)$  都无关。因此, 只要给出等分数  $n$ , 就能算出  $C_i^{(n)}$ , 从而写出相应的牛顿-柯特斯公式。表 8.2.1 列出了  $n=1 \sim 8$  的柯特斯系数, 从表中可以看出, 当  $n=8$  时, 柯特斯系数出现负数, 稳定性得不到保证, 所以一般采用  $n \leq 4$  的牛顿-柯特斯求积公式。

从表 8.2.1 还可以看出, 柯特斯系数对称, 并且其代数和为 1, 实际上, 由于式 (8.2.1) 具有  $n$  次代数精度, 特别取  $f(x)=1$ , 则得

$$b-a = \int_a^b 1 dx = (b-a) \sum_{i=0}^n C_i^{(n)} \times 1$$

从而有

$$\sum_{i=0}^n C_i^{(n)} = 1$$

当  $n=1$  时, 利用式 (8.2.2) 可求出  $C_0^{(1)} = C_1^{(1)} = \frac{1}{2}$ , 此时, 求积公式为

$$I(f) \approx T = \frac{b-a}{2} [f(a) + f(b)] \quad (8.2.3)$$

式 (8.2.3) 称为梯形求积公式。

当  $n=2$  时, 利用式 (8.2.2) 可求出  $C_0^{(2)} = \frac{1}{6}$ ,  $C_1^{(2)} = \frac{4}{6}$ ,  $C_2^{(2)} = \frac{1}{6}$ , 求积公式为

$$I(f) \approx S = \frac{b-a}{6} \left[ f(a) + 4f\left(\frac{b+a}{2}\right) + f(b) \right] \quad (8.2.4)$$

式 (8.2.4) 称为抛物线求积公式或辛普生 (Simpson) 求积公式。

同理, 可求出当  $n=4$  时的牛顿-柯特斯公式为

$$I(f) \approx C = \frac{b-a}{90} [7f(x_0) + 32f(x_1) + 12f(x_2) + 32f(x_3) + 7f(x_4)] \quad (8.2.5)$$

式 (8.2.5) 称为柯特斯求积公式, 式中  $x_i = a + ih$  ( $i = 0, 1, 2, 3, 4$ ),  $h = \frac{b-a}{4}$ 。

表 8.2.1 柯特斯系数

$n$	$C_i^{(n)}$								
1	$\frac{1}{2}$	$\frac{1}{2}$							
2	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$						
3	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$					
4	$\frac{7}{90}$	$\frac{16}{45}$	$\frac{2}{15}$	$\frac{16}{45}$	$\frac{7}{90}$				
5	$\frac{19}{288}$	$\frac{25}{96}$	$\frac{25}{144}$	$\frac{25}{144}$	$\frac{25}{96}$	$\frac{19}{288}$			
6	$\frac{41}{840}$	$\frac{9}{35}$	$\frac{9}{280}$	$\frac{34}{105}$	$\frac{9}{280}$	$\frac{9}{35}$	$\frac{41}{840}$		
7	$\frac{751}{17280}$	$\frac{3577}{17280}$	$\frac{1323}{17280}$	$\frac{2989}{17280}$	$\frac{2989}{17280}$	$\frac{1323}{17280}$	$\frac{3577}{17280}$	$\frac{751}{17280}$	
8	$\frac{989}{28350}$	$\frac{5888}{28350}$	$\frac{-928}{28350}$	$\frac{10496}{28350}$	$\frac{-4540}{28350}$	$\frac{10496}{28350}$	$\frac{-928}{28350}$	$\frac{5888}{28350}$	$\frac{989}{28350}$

牛顿-柯特斯求积公式 (8.2.1) 是在等距节点条件下的插值型求积公式, 因此, 至少具有  $n$  次代数精度, 但当  $n$  为偶数时, 此公式却可以达到  $n+1$  次代数精度。

**定理 8.2.1** 对于牛顿-柯特斯求积公式 (8.2.1), 当  $n$  为奇数时, 它至少具有  $n$  次代数精度; 当  $n$  为偶数时, 它却可以达到  $n+1$  次代数精度。

**证明** (只证明  $n$  为偶数的情形) 设被积函数  $f(x) = x^{n+1}$  ( $n$  为偶数), 由于  $f^{(n+1)}(x) = (n+1)!$ , 此时, 求积公式的余项是

$$R[f] = \int_a^b \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{j=0}^n (x - x_j) dx = \int_a^b \prod_{j=0}^n (x - x_j) dx$$

进行变换  $x = a + th, x_j = a + jh$ , 于是

$$R[f] = h^{n+2} \int_0^n \prod_{j=0}^n (t - j) dt$$

又因为  $n$  为偶数, 可设  $n = 2k$  ( $k$  为正整数) 并进行变换  $t = s + k$ , 则有

$$R[f] = h^{2k+2} \int_{-k}^k \prod_{j=0}^{2k} (s + k - j) ds$$

上式右端积分中的被积函数是关于  $s$  的函数, 设为  $G(s)$

$$\begin{aligned} G(s) &= \prod_{j=0}^{2k} (s + k - j) \\ &= (s + k)(s + k - 1) \cdots s(s - 1) \cdots (s - k + 1)(s - k) \\ &= \prod_{j=-k}^k (s - j) \end{aligned}$$

而

$$\begin{aligned} G(-s) &= \prod_{j=-k}^k (-s - j) = (-1)^{2k+1} \prod_{j=-k}^k (s + j) \\ &= -(s - k)(s - k + 1) \cdots (s - 1)s(s + 1) \cdots (s + k + 1)(s + k) \\ &= -\prod_{j=-k}^k (s - j) = -G(s) \end{aligned}$$

因此, 被积函数  $G(s)$  是奇函数, 所以它在对称区间  $[-k, k]$  内的定积分为 0, 从而有  $R[f] = 0$ 。即当  $n$  为偶数时, 式 (8.2.1) 对被积函数  $f(x) = x^{n+1}$  的余项为 0, 所以式 (8.2.1) 对  $f(x) = x^{n+1}$  精确成立, 因此求积公式 (8.2.1) 的代数精度至少是  $n+1$  次。证毕。

由定理 8.2.1 可知, 梯形公式 (8.2.3) 具有一次代数精度, 辛普生公式 (8.2.4) 具有 3 次代数精度, 柯特斯求积公式 (8.2.5) 具有 5 次代数精度。

**例 8.2.1** 验证辛普生公式

$$\int_a^b f(x) dx \approx \frac{b-a}{6} \left[ f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right]$$

和柯特斯公式

$$\begin{aligned} \int_a^b f(x) dx &\approx \frac{b-a}{90} [7f(x_0) + 32f(x_1) + 12f(x_2) + 32f(x_3) + 7f(x_4)] \\ (x_i &= a + i \frac{b-a}{4}, i = 0, 1, 2, 3, 4) \end{aligned}$$

分别具有 3 阶代数精度和 5 阶代数精度。

**解** 令  $x = \frac{b+a}{2} + t \frac{b-a}{2}$ , 则辛普生公式和柯特斯公式分别简化为

$$\begin{aligned} \int_{-1}^1 f(x) dx &\approx \frac{1}{3} [f(-1) + 4f(0) + f(1)] \\ \int_{-1}^1 f(x) dx &\approx \frac{1}{45} \left[ 7f(-1) + 32f\left(-\frac{1}{2}\right) + 12f(0) + 32f\left(\frac{1}{2}\right) + 7f(1) \right] \end{aligned}$$

可直接将  $f(x) = 1, t, t^2, t^3, t^4, t^5$  代入验证, 利用对称性, 容易检验这两个求积公式的代数精度分别为 3 次和 5 次。

## 8.2.2 牛顿-柯特斯公式的误差估计

牛顿-柯特斯求积公式的余项可用式 (8.1.7) 表示, 当  $n=1$  时, 有

$$R_1(f) = I(f) - T_1 = \frac{1}{2} \int_a^b f''(\xi)(x-a)(x-b)dx$$

由于  $(x-a)(x-b) \leq 0$ ，在  $[a, b]$  内不变号，由定理 2.1.8 可知，存在  $\eta \in (a, b)$ ，使

$$R_1(f) = \frac{f''(\eta)}{2} \int_a^b (x-a)(x-b)dx = -\frac{f''(\eta)}{12}(b-a)^3 \quad (\eta \in (a, b)) \quad (8.2.6)$$

式 (8.2.6) 为梯形公式 (8.2.3) 的截断误差。

当  $n=2$  时，有

$$R_2(f) = \int_a^b \frac{f'''(\xi)}{3!} (x-a)(x-\frac{a+b}{2})(x-b)dx$$

由于  $(x-a)\left(x-\frac{a+b}{2}\right)(x-b)$  在区间  $[a, b]$  内不保号，即符号可正可负，故不能直接应用定理 2.1.8。

但因为辛普生公式 (8.2.4) 具有 3 次代数精度，它对于满足条件

$$\begin{cases} H(a) = f(a) & H(b) = f(b) \\ H(c) = f(c) & H'(c) = f'(c) \end{cases} \quad (c = \frac{a+b}{2}) \quad (8.2.7)$$

的 3 次插值多项式  $H(x)$  能准确成立，故有

$$\int_a^b H(x)dx = \frac{b-a}{6} [H(a) + 4H(c) + H(b)]$$

而利用插值条件式 (8.2.7) 可知，积分值  $\int_a^b H(x)dx$  实际上等于辛普生公式求得的积分值，从而有

$$R_2[f] = I(f) - S_1 = \int_a^b [f(x) - H(x)]dx$$

再利用埃尔米特插值的余项公式得

$$R_2[f] = \int_a^b \frac{f^{(4)}(\xi)}{4} (x-a)(x-c)^2(x-b)dx$$

由于  $(x-a)(x-c)^2(x-b)$  在  $[a, b]$  内保号（非正），因此利用积分中值定理得到

$$\begin{aligned} R_2[f] &= \frac{f^{(4)}(\eta)}{4!} \int_a^b (x-a)(x-c)^2(x-b)dx \\ &= -\frac{b-a}{180} \left(\frac{b-a}{2}\right)^4 f^{(4)}(\eta) \quad (\eta \in (a, b)) \end{aligned} \quad (8.2.8)$$

式 (8.2.8) 为辛普生公式 (8.2.4) 的截断误差。

利用类似的方法，可以求出柯特斯求积公式 (8.2.5) 的截断误差为

$$R_4[f] = I(f) - C_1 = -\frac{2(b-a)}{945} \left(\frac{b-a}{4}\right)^6 f^{(6)}(\eta) \quad (\eta \in (a, b)) \quad (8.2.9)$$

**例 8.2.2** 如果  $f''(x) > 0$ ，证明用梯形公式计算积分  $I = \int_a^b f(x)dx$  所得结果比准确值大，并

说明其几何意义。

**证明** 由梯形公式的余项

$$R(f) = -\frac{(b-a)^3}{12} f''(\eta) \quad (\eta \in (a, b))$$

可知，若  $f''(x) > 0$ ，则  $R(f) < 0$ ，从而

$$\int_a^b f(x)dx = T + R(f) < T$$

即用梯形公式计算积分所得结果比准确值大。其几何意义为  $f''(x) > 0$ ，故  $f(x)$  为下凸函数，梯



形面积大于曲边梯形面积。

现在, 讨论牛顿-柯特斯求积公式的稳定性和收敛性的问题。

稳定性问题就是研究计算和式

$$I_n(f) = (b-a) \sum_{i=0}^n [C_i^{(n)} f(x_i)]$$

当  $f(x_i)$  有误差  $\delta_i$  (此时设  $f(x_i) \approx \tilde{f}_i$ ) 时,  $I_n(f)$  的误差是否增长。

现设  $f(x_i) \approx \tilde{f}_i$ , 误差  $\delta_i = |f(x_i) - \tilde{f}_i|$  ( $i = 0, 1, 2, \dots, n$ )。

**定义 8.2.1 (数值稳定性)** 对任意  $\varepsilon > 0$ , 若存在  $\delta > 0$ , 只要  $\delta_i = |f(x_i) - \tilde{f}_i| \leq \delta$  ( $i = 0, 1, 2, \dots, n$ ),

就有

$$|I_n(f) - I_n(\tilde{f})| \leq \varepsilon$$

则称被积公式 (8.2.1) 是数值稳定的。

定义 8.2.1 表明只要被积函数  $f(x)$  的误差  $\delta_i$  充分小, 积分和式  $I_n(f)$  的误差就任意小, 这说明式 (8.2.1) 就是数值稳定的。

**定理 8.2.2** 若求积公式 (8.2.1) 的系数  $C_i^{(n)} > 0$  ( $i = 0, 1, 2, \dots, n$ ), 则求积公式 (8.2.1) 是稳定的。

**证明** 由于

$$\begin{aligned} C_i^{(n)} &> 0 \quad (i = 0, 1, 2, \dots, n) \\ |f(x_i) - \tilde{f}_i| &\leq \delta \quad (i = 0, 1, 2, \dots, n) \end{aligned}$$

故有

$$\begin{aligned} |I_n(f) - I_n(\tilde{f})| &= \left| (b-a) \sum_{i=0}^n [C_i^{(n)} (f(x_i) - \tilde{f}_i)] \right| \leq \delta (b-a) \sum_{i=0}^n C_i^{(n)} \\ &= \delta (b-a) \end{aligned}$$

于是, 对任意的  $\varepsilon > 0$ , 存在  $\delta = \frac{\varepsilon}{b-a}$ , 只要  $\delta_i = |f(x_i) - \tilde{f}_i| \leq \delta$ , 就有

$$|I_n(f) - I_n(\tilde{f})| \leq \delta (b-a) \leq \varepsilon$$

故求积公式 (8.1.3) 是数值稳定的。

从表 8.2.1 可以看出, 当  $n \geq 8$  时, 柯特斯系数出现负值, 此时牛顿-柯特斯求积公式是不稳定的。另外, 可以证明, 并非一切连续函数  $f(x)$ , 当  $n \rightarrow \infty$  时, 都有  $R_n[f] \rightarrow 0$ , 即牛顿-柯特斯求积公式的收敛性没有保证, 因此, 在实际计算时, 很少使用高阶的牛顿-柯特斯求积公式。

## 8.3 复合求积公式

从梯形公式、辛普生公式以及柯特斯求积公式的余项式 (8.2.6)、式 (8.2.8) 和式 (8.2.9)

可以看出, 数值求积的误差除了与被积函数有关外, 还与积分区间的长度  $(b-a)$  有关, 积分区间越小, 求积公式的截断误差也就越小。因此, 在求积分时, 常把积分区间等分成若干小区间, 在每个小区间内采用次数不高的求积公式, 如梯形公式、辛普生公式, 然后再把它们加起来, 得到整个区间内的求积公式, 这就是复合求积公式的基本思想。

### 8.3.1 复合梯形求积公式

将  $[a, b]$  区间  $n$  等分, 记分点为

$$x_i = a + ih \quad (h = \frac{b-a}{n}, \quad i = 0, 1, \dots, n)$$

并在每个小区间  $[x_i, x_{i+1}]$  内应用梯形公式 (8.2.3), 得

$$\begin{aligned} \int_a^b f(x) dx &= \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x) dx \approx \sum_{i=0}^{n-1} \frac{h}{2} [f(x_i) + f(x_{i+1})] \\ &= \frac{h}{2} \left[ f(a) + 2 \sum_{i=1}^{n-1} f(x_i) + f(b) \right] \end{aligned}$$

记

$$T_n = \frac{h}{2} \left[ f(a) + 2 \sum_{i=1}^{n-1} f(x_i) + f(b) \right] \quad (8.3.1)$$

式 (8.3.1) 称为复合梯形公式, 下标  $n$  表示将区间  $n$  等分。若把区间  $2n$  等分, 在每个小区间内仍用梯形求积公式, 则可以得到  $T_{2n}$ 。  $T_n$  与  $T_{2n}$  之间的关系为

$$T_{2n} = \frac{1}{2} (T_n + H_n) \quad (8.3.2)$$

式中  $H_n = h \sum_{i=1}^n f(x_{i+\frac{1}{2}})$ ,  $x_{i+\frac{1}{2}}$  为  $[x_i, x_{i+1}]$  的中点, 即  $x_{i+\frac{1}{2}} = x_i + \frac{1}{2}h$ , 根据定积分的定义可知

$$\lim_{\substack{n \rightarrow \infty \\ h \rightarrow 0}} T_n = \lim_{h \rightarrow 0} \frac{1}{2} \left[ \sum_{i=0}^{n-1} f(x_i) h + \sum_{i=1}^n f(x_i) h \right] = \int_a^b f(x) dx = I(f)$$

故复合梯形公式 (8.3.1) 是收敛的, 且式中的系数  $A_i > 0 (i = 0, 1, 2, \dots, n)$ , 因此式 (8.3.1) 也是稳定的。

复合梯形公式 (8.3.1) 的截断误差可由式 (8.2.6) 得到

$$\begin{aligned} R_n(f) &= I(f) - T_n = \sum_{i=0}^{n-1} \left[ -\frac{h^3}{12} f''(\eta_i) \right] \\ &= -\frac{h^2}{12} (b-a) \frac{1}{n} \sum_{i=0}^{n-1} f''(\eta_i) \quad (\eta_i \in (x_i, x_{i+1})) \end{aligned}$$

设  $f''(x)$  在  $[a, b]$  内连续, 根据连续函数的性质, 函数  $f(x)$  在  $[a, b]$  内必有一点  $\eta$ , 使

$$f''(\eta) = \frac{1}{n} \sum_{i=0}^{n-1} f''(\eta_i)$$

于是有

$$R_n(f) = -\frac{b-a}{12} h^2 f''(\eta) \quad (\eta \in (a, b)) \quad (8.3.3)$$

式 (8.3.3) 称为复合梯形公式的截断误差, 误差阶为  $R_n(f) = O(h^2)$ 。

### 8.3.2 复合辛普生求积公式

在每个小区间  $[x_i, x_{i+1}]$  内, 用辛普生公式 (8.2.4) 得

$$\begin{aligned}\int_a^b f(x)dx &= \sum_{i=0}^{n-1} \frac{h}{6} \left[ f(x_i) + 4f(x_{i+\frac{1}{2}}) + f(x_{i+1}) \right] \\ &= \frac{h}{6} \left[ f(a) + 4 \sum_{i=0}^{n-1} f(x_{i+\frac{1}{2}}) + 2 \sum_{i=1}^{n-1} f(x_i) + f(b) \right]\end{aligned}$$

记

$$S_n = \frac{h}{6} \left[ f(a) + 4 \sum_{i=0}^{n-1} f(x_{i+\frac{1}{2}}) + 2 \sum_{i=1}^{n-1} f(x_i) + f(b) \right] \quad (8.3.4)$$

式中  $x_{i+\frac{1}{2}}$  为  $[x_i, x_{i+1}]$  的中点, 即  $x_{i+\frac{1}{2}} = x_i + \frac{1}{2}h$ 。

式 (8.3.4) 称为复合辛普生公式, 其余项可由式 (8.2.8) 得到

$$\begin{aligned}R_n(f) &= I(f) - S_n = -\frac{h}{180} \left( \frac{h}{2} \right)^4 \sum_{i=0}^{n-1} f^{(4)}(\eta_i) \quad (\eta_i \in (x_i, x_{i+1})) \\ &= -\frac{b-a}{180} \left( \frac{h}{2} \right)^4 f^{(4)}(\eta) \quad (\eta \in (a, b))\end{aligned}$$

记

$$R_n(f) = -\frac{b-a}{180} \left( \frac{h}{2} \right)^4 f^{(4)}(\eta) \quad (\eta \in (a, b)) \quad (8.3.5)$$

式 (8.3.5) 为复合辛普生公式的截断误差, 误差阶为  $O(h^4)$ 。

可以证明

$$\lim_{n \rightarrow \infty} S_n = \int_a^b f(x)dx$$

这说明式 (8.3.4) 是收敛的, 并且由于式 (8.3.4) 中的  $A_i > 0$  ( $i = 0, 1, 2, \dots, n$ ), 因此式 (8.3.4) 也是稳定的。

**例 8.3.1** 用  $n=8$  的复合梯形公式及  $n=4$  的复合辛普生公式, 计算积分  $I_n = \int_0^1 \frac{\sin x}{x} dx$ , 并估计误差 (精确值  $I=0.9460831$ )。

**解** 首先计算出所需各节点的函数值,  $n=8$  时,  $h = \frac{1}{8} = 0.125$ , 各点函数值见表 8.3.1。

表 8.3.1 各点函数值

$k$	$x_k$	$f(x_k)$	$k$	$x_k$	$f(x_k)$
0	0	1.000 000 0	5	0.625	0.936 155 6
1	0.125	0.997 397 8	6	0.75	0.908 851 6
2	0.25	0.989 615 8	7	0.875	0.877 192 5
3	0.375	0.976 726 7	8	1	0.841 470 9
4	0.5	0.958 851 0			

由式 (8.3.1) 得

$$T_8 = \frac{1}{16}(f(0) + 2\sum_{i=1}^7 f(x_i) + f(1)) = 0.94569081$$

由式 (8.3.4) 得

$$S_4 = \frac{1}{24}[(f(0) + f(1) + 2(f(0.25) + f(0.5) + f(0.75)) + 4(f(0.125) + f(0.375) + f(0.625) + f(0.875)))] \\ = 0.94608325$$

为了估计误差, 要求  $f(x) = \frac{\sin x}{x}$  的高阶导数, 由于

$$f(x) = \frac{\sin x}{x} = \int_0^1 \cos(xt) dt$$

因此

$$f^{(n)} = \int_0^1 \frac{d^n}{dx^n} \cos(xt) dt = \int_0^1 t^n \cos(xt + \frac{n\pi}{2}) dt$$

所以

$$|f^{(n)}(x)| \leq \int_0^1 t^n \left| \cos(xt + \frac{n\pi}{2}) \right| dt \leq \int_0^1 t^n dt = \frac{1}{n+1}$$

对复合梯形公式, 由式 (8.3.3) 得

$$R_8(f) = |I(f) - T_8| = \left| -\frac{1}{12} h^2 f''(\eta) \right| \leq \frac{1}{12} \times \left( \frac{1}{8} \right)^2 \times \frac{1}{3} = 0.000434 \\ \leq 0.0005 = \frac{1}{2} \times 10^{-3}$$

对复合辛普生公式, 由式 (8.3.5) 得

$$|R_4(f)| = |I(f) - S_4| \leq \left| \frac{1}{180} \times \left( \frac{1}{16} \right) \times \frac{1}{5} \right| = 0.271 \times 10^{-6} \leq \frac{1}{2} \times 10^{-6}$$

所以  $T_8$  有 3 位有效数字,  $S_4$  有 6 位有效数字。

从例 8.3.1 可以看出, 计算  $T_8$  和  $S_4$  都需要 9 个点上的函数值, 计算工作量基本相同, 然而精度却差别很大, 这说明在对积分区间进行同样分割或利用同样个数的函数值的条件下, 复合辛普生公式比复合梯形公式的计算精度高, 因此, 在实际计算时, 复合辛普生公式应用较为普遍。

为了方便编程实现, 可将式 (8.3.4) 改写成如下形式

$$S_n = \frac{h}{6} \left\{ f(a) - f(b) + \sum_{i=1}^n \left[ 4f\left(f_{x_i + \frac{1}{2}}\right) + 2f(x_i) \right] \right\} \quad (8.3.6)$$

根据式 (8.3.6) 写出复合辛普生公式的算法框图, 如图 8.3.1 所示。

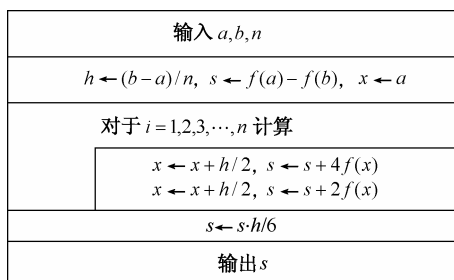


图 8.3.1 复合辛普生公式的算法框图

**例 8.3.2** 计算积分  $I = \int_0^1 e^x dx$ , 若用复合梯形公式, 应将  $[0,1]$  区间几等分, 才能使截断误差不超过  $\frac{1}{2} \times 10^{-5}$ ? 若改用复合辛普生公式, 应将  $[0,1]$  区间几等分, 才能达到同样的精度?

**解** 由于  $f(x) = e^x, f''(x) = e^x, f^{(4)}(x) = e^x, b-a=1$ , 根据  $T_n$  的误差公式 (8.3.3) 得

$$|R_n(f)| = \left| -\frac{b-a}{12} h^2 f''(\eta) \right| \leq \frac{1}{12} \times \left( \frac{1}{n} \right)^2 e \leq \frac{1}{2} \times 10^{-5}$$

即  $n^2 \geq \frac{1}{6} e \times 10^5, n \geq 212.85$ , 取  $n = 213$ 。所以, 需要将  $[0,1]$  区间 213 等分, 即必须用  $n = 213$  复

合梯形公式计算, 误差才不超过  $\frac{1}{2} \times 10^{-5}$ 。

根据  $S_n$  的误差公式 (8.3.5) 得

$$|R_n(f)| = \left| -\frac{b-a}{180} \left( \frac{h}{2} \right)^4 f^{(4)}(\eta) \right| \leq \frac{1}{2880} \times \left( \frac{1}{n} \right)^4 \times e \leq \frac{1}{2} \times 10^{-5}$$

即  $n^2 \geq \frac{e}{144}, n \geq 3.707$ , 取  $n = 4$ 。所以, 需要将  $[0,1]$  区间 8 等分, 即必须用  $n = 4$  的复合辛普生公式计算, 误差才不超过  $\frac{1}{2} \times 10^{-5}$ 。

## 8.4 外推算法与龙贝格算法

### 8.4.1 变步长的求积公式

在利用复合求积公式计算积分之前, 必须给出适当的步长 ( $h = \frac{b-a}{n}$ ), 步长如果太大, 精度难以保证; 步长如果太小, 则会导致计算量的增大。在计算之前给出一个恰当的步长, 往往非常困难, 因此, 在实际计算中, 常采用变步长的方法, 其基本思想是: 在步长逐次折半的过程中, 反复用复合求积公式进行计算, 直到步长折半前后的两次积分值之差的绝对值  $|I_{2n}(f) - I_n(f)|$  小于允许的精度为止, 并取  $I_{2n}(f)$  作为所求积分的近似值。

以复合梯形公式为例, 将  $[a,b]$  区间  $n$  等分, 步长  $h = \frac{b-a}{n}$ , 则

$$T_n = T(h) = \frac{h}{2} \left[ f(a) + 2 \sum_{i=1}^{n-1} f(x_i) + f(b) \right] \quad (8.4.1)$$

若将  $[a,b]$  区间  $2n$  等分, 步长为原来的一半, 变为  $\frac{h}{2} = \frac{b-a}{2n}$ , 则

$$\begin{aligned} T_{2n} &= T\left(\frac{h}{2}\right) = \frac{1}{2} \left(\frac{h}{2}\right) \left[ f(a) + 2 \sum_{i=1}^{n-1} f(x_i) + 2 \sum_{i=0}^{n-1} f(x_{i+\frac{1}{2}}) + f(b) \right] \\ &= \frac{1}{2} T_n + \frac{h}{2} \sum_{i=0}^{n-1} f(x_{i+\frac{1}{2}}) \end{aligned} \quad (8.4.2)$$

因为复合梯形公式是收敛的, 所以用逐次折半的方法构造出的近似值序列

$$T(h), T\left(\frac{h}{2}\right), T\left(\frac{h}{2^2}\right), \dots, T\left(\frac{h}{2^n}\right), \dots$$

收敛于精确值  $T(0)$ 。

**例 8.4.1** 用变步长的梯形法求例 8.3.1 的积分  $I(f) = \int_0^1 \frac{\sin x}{x} dx$  的近似值。

**解** 补充定义  $f(0) = 1$ ，计算得  $f(1) = 0.8414709$ ，所以

$$T_1 = \frac{1}{2}[f(0) + f(1)] = 0.92073545$$

将  $[0, 1]$  区间 2 等分，计算得  $f\left(\frac{1}{2}\right) = 0.9588510$ ，利用式 (8.4.2) 得

$$T_2 = \frac{1}{2}T_1 + \frac{1}{2}f\left(\frac{1}{2}\right) = 0.9397933$$

将  $[0, 1]$  区间 4 等分，计算得  $f\left(\frac{1}{4}\right) = 0.9896158, f\left(\frac{3}{4}\right) = 0.9088516$ ，利用式 (8.4.2) 得

$$T_4 = \frac{1}{2}T_2 + \frac{1}{4}\left[f\left(\frac{1}{4}\right) + f\left(\frac{3}{4}\right)\right] = 0.94451347$$

这样，将区间逐次二分，步长逐次折半，计算结果见表 8.4.1（表中， $k$  为二分次数，区间份数为  $n = 2^k$ ）。

表 8.4.1 步长逐次折半的计算结果

$k$	0	1	2	3	4	5
$T_{2^k}$	0.920 735 5	0.939 793 3	0.944 513 5	0.945 690 9	0.945 985 0	0.946 058 6
$k$	6	7	8	9	10	11
$T_{2^k}$	0.946 076 9	0.946 081 5	0.946 082 7	0.946 083 0	0.946 083 0	0.946 083 1

积分  $I(f)$  的精确值为 0.9460831，用变步长的复合梯形公式步长逐次折半 11 次（也就是将  $[0, 1]$  区间 2048 等分）以后，才得到了这个结果，这说明，收敛速度非常缓慢。

## 8.4.2 外推算法

在数值计算中，常常利用一个序列  $T_1, T_2, \dots, T_n, \dots$ ，去逼近准确解  $T$ ，序列  $\{T_i\} (i = 1, 2, \dots)$  又经常与区间的步长有关。例如用变步长的复合梯形公式产生一个序列

$$T(h), T\left(\frac{h}{2}\right), \dots, T\left(\frac{h}{2^n}\right), \dots$$

去逼近准确值  $T(0)$ 。

由于这一序列收敛的速度很慢，因此，如何提高收敛速度以节省计算量便成了一个最实际的问题。这一问题，可以推广到更一般的情况。

给定一个收敛于  $f(0)$  的序列  $f(h), f\left(\frac{h}{2}\right), \dots$ ，能否在此基础上构造一个新的序列，使其更快的收敛于  $f(0)$ 。这在某些条件下是可以实现的，例如利用泰勒展开式。

$$f(h) = f(0) + hf'(0) + \frac{h^2}{2!}f''(0) + \frac{h^3}{3!}f'''(0) + \dots \quad (8.4.3)$$

$$f\left(\frac{h}{2}\right) = f(0) + \frac{h}{2}f'(0) + \frac{1}{2!}\left(\frac{h}{2}\right)^2 f''(0) + \frac{1}{3!}\left(\frac{h}{2}\right)^3 f'''(0) + \cdots \quad (8.4.4)$$

如果  $f'(0) \neq 0$ ，那么  $f(h), f\left(\frac{h}{2}\right)$  逼近  $f(0)$  的阶都是  $O(h)$ 。

现在将式 (8.4.4) 乘以 2，再减去式 (8.4.3) 得

$$f_1(h) = 2f\left(\frac{h}{2}\right) - f(h) = f(0) - \frac{h^2}{4}f''(0) - \frac{h^3}{8}f'''(0) + \cdots$$

如果  $f''(0) \neq 0$ ，那么  $2f\left(\frac{h}{2}\right) - f(h)$  逼近  $f(0)$  的误差阶为  $O(h^2)$ 。即新序列  $f_1(h) = 2f\left(\frac{h}{2}\right) - f(h)$  有可能比旧序列  $f(h)$  更快地收敛于  $f(0)$ 。

可以继续从序列  $f_1(h)$  再产生出新的序列  $f_2(h)$ ，使其更快地收敛于  $f(0)$ ，由于

$$\begin{aligned} f_1(h) &= f(0) - \frac{h^2}{4}f''(0) - \frac{h^3}{8}f'''(0) + \cdots \\ f_1\left(\frac{h}{2}\right) &= f(0) - \frac{h^2}{16}f''(0) - \frac{h^3}{64}f'''(0) + \cdots \end{aligned}$$

将后式乘以 4，再减去前式，有

$$h_2(f) = \frac{4f_1\left(\frac{h}{2}\right) - f_1(h)}{3} = f(0) + \frac{h^3}{48}f'''(0) + \cdots$$

如果  $f'''(0) \neq 0$ ，则序列  $\{f_2(h)\}$  逼近  $f(0)$  的误差阶为  $O(h^3)$ 。

上述推理过程是利用若干近似值推算出更精确的近似值的加速收敛的方法，这种方法称为外推算法，外推算法在数值积分和数值微分等数值计算中都有广泛的应用。

### 8.4.3 龙贝格求积公式

用变步长的求积公式，可以根据精度的要求，在计算过程中适当调整步长，使其计算结果逐步逼近精确值，但是近似值序列收敛于积分精确值的速度较慢，现在利用外推算法的原理，来研究提高收敛速度的方法。

由复合梯形法的误差公式 (8.3.3) 知，积分值  $T_n$  的截断误差为

$$I - T_n = -\frac{b-a}{12}h^2 f''(\eta_1)$$

它与  $h^2$  成正比。若把每个小区间  $[x_i, x_{i+1}]$  ( $i=0, 1, 2, \dots, n-1$ ) 再对分，即将  $[a, b]$  区间  $2n$  等分，则截断误差为

$$I - T_{2n} = -\frac{b-a}{12}\left(\frac{h}{2}\right)^2 f''(\eta_2)$$

如果  $f''(x)$  在积分区间  $[a, b]$  内变化不大，即有

$$f''(\eta_1) \approx f''(\eta_2)$$

则由上述两个误差公式可得

$$\frac{I - T_n}{I - T_{2n}} \approx 4$$

从而

$$I \approx \frac{4}{3}T_{2n} - \frac{1}{3}T_n = \tilde{T}$$

计算结果  $\tilde{T}$  应该比  $T_n$  和  $T_{2n}$  更精确。当  $n=1$  时, 直接计算有

$$\begin{aligned}\tilde{T} &= \frac{4}{3}T_2 - \frac{1}{3}T_1 = \frac{4}{3} \times \frac{1}{2} \times \frac{b-a}{2} \times [f(a) + 2 \times f(\frac{a+b}{2}) + f(b)] - \frac{1}{3} \times \frac{b-a}{2} \times [f(a) + f(b)] \\ &= \frac{b-a}{6} \times [f(a) + 4 \times f(\frac{a+b}{2}) + f(b)]\end{aligned}$$

这正好是在  $[a, b]$  区间内应用辛普生求积公式的结果, 即

$$S_1 = \frac{4}{3}T_2 - \frac{1}{3}T_1 = \frac{4}{4-1}T_2 - \frac{1}{4-1}T_1$$

进一步, 可以验证

$$S_2 = \frac{4}{4-1}T_4 - \frac{1}{4-1}T_2$$

这正好是将  $[a, b]$  区间 2 等分后, 在每个小区间  $[x_i, x_{i+1}]$  ( $i=0, 1$ ) 内应用辛普生求积公式的结果。

一般地

$$S_n = \frac{4}{4-1}T_{2n} - \frac{1}{4-1}T_n \quad (8.4.5)$$

$S_n$  是将  $[a, b]$  区间  $n$  等分后, 在每个小区间  $[x_i, x_{i+1}]$  ( $i=0, 1, 2, \dots, n-1$ ) 内应用辛普生求积公式的结果, 这说明, 用复合梯形法二分前后的两个积分近似值  $T_n$  和  $T_{2n}$  按式 (8.4.5) 计算, 就可以得到精度较高的辛普生求积公式的近似值  $S_n$ , 从而加速了逼近的效果, 所以称式 (8.4.5) 为梯形加速公式。

由复合辛普生法的误差公式 (8.3.5) 可知, 积分值  $S_n$  的截断误差为

$$I - S_n = -\frac{b-a}{180} \left(\frac{h}{2}\right)^4 f^{(4)}(\eta_1)$$

它与  $h^4$  成比例, 因此, 如果  $f^{(4)}(x)$  在  $[a, b]$  区间内变化不大, 用同样的处理方法, 则将步长折半后, 误差将减至原有误差的  $\frac{1}{16}$ , 即有

$$\frac{I - S_{2n}}{I - S_n} \approx \frac{1}{16}$$

从而有

$$I \approx \frac{16}{15}S_n - \frac{1}{15}S_{2n}$$

可以验证, 上式右端的值, 是将  $[a, b]$  区间  $n$  等分后, 在每个小区间  $[x_i, x_{i+1}]$  ( $i=0, 1, 2, \dots, n-1$ ) 内用柯特斯公式得到的, 具有更高精度的复合柯特斯公式的积分值  $C_n$ , 即有

$$C_n = \frac{4^2}{4^2-1}S_{2n} - \frac{1}{4^2-1}S_n \quad (8.4.6)$$

式 (8.4.6) 称为辛普生加速公式。

重复同样的方法, 依据复合柯特斯公式的误差公式, 可进一步得出公式

$$R_n = \frac{4^3}{4^3-1}C_{2n} - \frac{1}{4^3-1}C_n \quad (8.4.7)$$

由式 (8.4.7) 计算出的序列  $R_1, R_2, \dots, R_n, \dots$  可以预测, 它逼近积分精确值的速度会得到很大提



高。式 (8.4.7) 称为柯特斯加速公式。

按照上述思路, 还可以构造出新的求积公式, 且右端有两个系数, 分别为  $\frac{4^m}{4^m-1}$  和  $\frac{1}{4^m-1}$ , 但当  $m \geq 4$  时, 第一个系数接近于 1, 第二个系数的绝对值很小, 接近于 0, 因此, 这样组合的新公式与前一个公式的计算结果没有多大区别, 反而增加了计算的工作量。因此, 在实际计算时, 只计算到式 (8.4.7) 为止。通常, 将变步长梯形法积分的近似值利用式 (8.4.5)、式 (8.4.6) 和式 (8.4.7) 这 3 个加速公式加工成精度更高的积分近似值的方法称为龙贝格 (Romberg) 求积算法。特别地, 式 (8.4.7) 称为龙贝格求积公式。要计算  $R_1$ , 需要先计算  $C_2, S_4, T_8$ , 即将  $[a, b]$  区间 8 等分, 计算 9 个节点的函数值, 但从公式的推导过程可知,  $R_1$  的误差阶为  $O(h^7)$ , 代数精度为 7 次, 因此, 它已不属于插值型求积公式, 从而也不属于牛顿-柯特斯求积公式的范畴。

用龙贝格方法计算积分的步骤如下。

① 准备初值, 先用梯形公式计算积分近似值。

$$T_1 = \frac{b-a}{2} [f(a) + f(b)]$$

② 按变步长梯形公式计算积分近似值。令

$$h = \frac{b-a}{2^i} \quad (i=0, 1, 2, \dots)$$

计算

$$T_{2n} = \frac{1}{2} T_n + \frac{h}{2} \sum_{i=0}^{n-1} f\left(x_{i+\frac{1}{2}}\right) \quad (n=2^i)$$

③ 按加速公式求积分。

为便于编程, 写为下列形式:

$$\text{梯形加速 } S_n = T_{2n} + (T_{2n} - T_n) / 3$$

$$\text{辛普生加速 } C_n = S_{2n} + (S_{2n} - S_n) / 15$$

$$\text{柯特斯加速 } R_n = C_{2n} + (C_{2n} - C_n) / 63$$

④ 精度控制。

当  $|R_{2n} - R_n| < \varepsilon$  ( $\varepsilon$  为精度) 时, 终止计算, 并取  $R_{2n}$  为近似值, 否则, 将步长折半, 转②执行。

实际计算时的加工流程如图 8.4.1 所示。

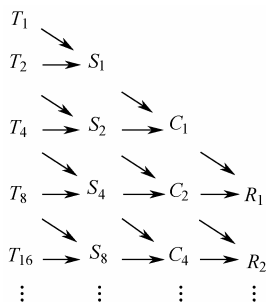


图 8.4.1 龙贝格算法加工流程

龙贝格算法框图如图 8.4.2 所示。

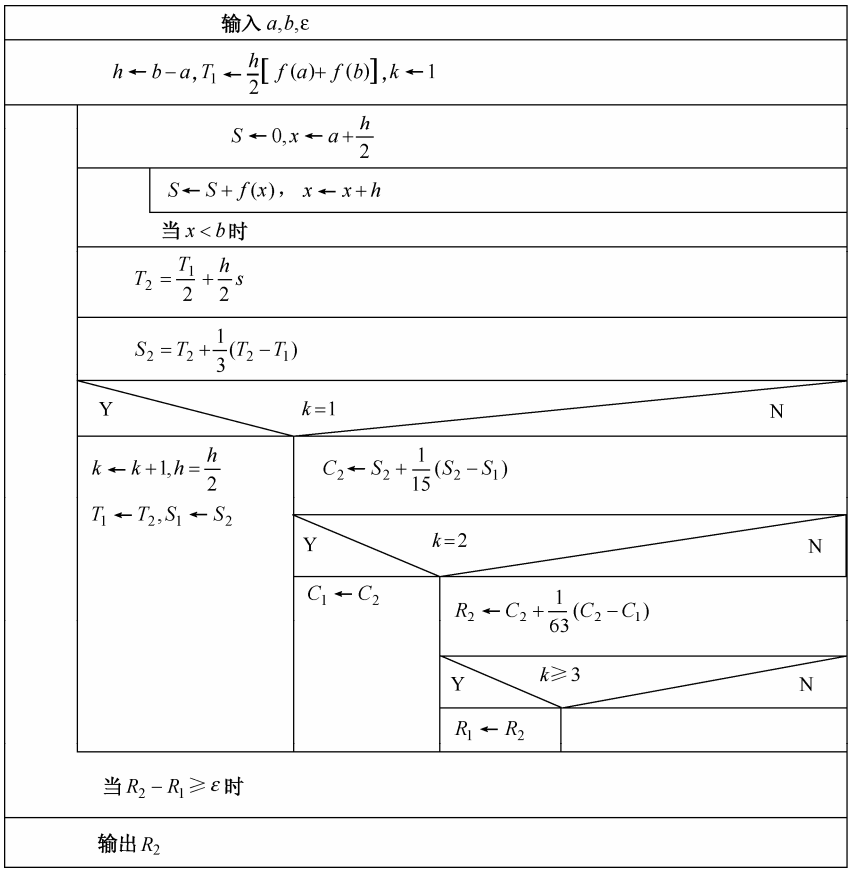


图 8.4.2 龙贝格算法框图

例 8.4.2 用龙贝格算法加工例 8.4.1 得到的近似值。

解 计算结果见表 8.4.2，其中  $k$  表示二分次数。

表 8.4.2 例 8.4.2 的计算结果

$k$	$T_2^k$	$S_{2^{k-1}}$	$C_{2^{k-2}}$	$R_{2^{k-3}}$
0	0.920 735 5			
1	0.939 793 3	0.946 145 9	0.946 083 0	
2	0.944 513 5	0.946 086 9	0.946 083 1	0.946 083 1
3	0.945 690 9	0.946 083 4		

从表 8.4.1 的结果可以看出，用二分 3 次的精度很低的数据通过 3 次加速得到了例 8.4.1 需要二分 11 次才能求得的结果，同时，加速过程的计算量只需几次算术运算，而不要求函数值，因此算术运算可以忽略不计。由此可见，龙贝格加速过程的效果是非常明显的。

## 8.5 高斯求积公式

### 8.5.1 高斯点与高斯求积公式

由定理 8.1.2 可知, 插值型求积公式的代数精度与求积节点的个数有关, 具有  $n+1$  个节点的插值型求积公式至少具有  $n$  次代数精度。不仅如此, 代数精度与节点的选取也有关, 在构造牛顿-柯特斯求积公式时, 为了简化处理过程, 限定用等分节点作为求积节点, 这样做, 虽然公式确实得到了简化, 但同时也限制了公式的代数精度。

设积分限  $a=-1, b=1$ , 本节讨论如下求积公式

$$\int_{-1}^1 f(x)dx = \sum_{i=0}^n A_i f(x_i) \quad (8.5.1)$$

对于任意积分区间  $[a, b]$ , 通过变换

$$x = \frac{b-a}{2}t + \frac{a+b}{2} \quad (8.5.2)$$

可以转化到区间  $[-1, 1]$  内, 这时

$$\int_a^b f(x)dx = \frac{b-a}{2} \int_{-1}^1 f\left(\frac{b-a}{2}t + \frac{a+b}{2}\right)dt \quad (8.5.3)$$

此时, 求积公式写为

$$\int_a^b f(x)dx = \frac{b-a}{2} \sum_{i=0}^n A_i f\left(\frac{a+b}{2} + \frac{b-a}{2}t_i\right) \quad (8.5.4)$$

适当选取式 (8.5.1) 中的节点位置  $x_i$  和求积系数  $A_i$ , 可以使其代数精度达到最高。

**定理 8.5.1** 插值型求积公式 (8.5.1) 具有的代数精度最高不超过  $2n+1$  次。

**证明** 由代数精度的定义, 只要找到一个  $2n+2$  次的多项式, 使求积公式 (8.5.1) 不能精确成立即可。设

$$\omega_{n+1}(x) = (x-x_0)(x-x_1)\cdots(x-x_n)$$

则  $\omega_{n+1}(x)$  为  $n+1$  次多项式, 取  $p(x) = [\omega_{n+1}(x)]^2$ , 则  $p(x)$  为  $2n+2$  次多项式, 且只在  $x_0, x_1, \dots, x_n$  处  $p(x)$  为 0。因此

$$\int_{-1}^1 p(x)dx = \int_{-1}^1 [\omega_{n+1}(x)]^2 dx > 0$$

而  $\sum_{i=0}^n A_i f(x_i) = 0$ , 因此, 当取  $f(x) = (\omega_{n+1}(x))^2$  时, 求积公式 (8.5.1) 不成立, 所以, 其代数精度不可能超过  $2n+1$  次。

**定义 8.5.1 (高斯点)** 若有一组节点  $x_0, x_1, \dots, x_n \in [-1, 1]$ , 使插值型求积公式 (8.5.1) 具有  $2n+1$  次代数精度, 则称此组节点为高斯点, 并称相应的求积公式 (8.5.1) 为高斯求积公式。

由于式 (8.5.1) 是插值型求积公式, 只要节点  $x_i$  确定了, 求积系数  $A_i$  也随之确定, 因此, 解决问题的关键在于节点  $x_i$  的选取, 即构造高斯求积公式的关键是求高斯点。

**定理 8.5.2** 插值型求积公式 (8.5.1) 中, 节点  $x_i (i=0, 1, \dots, n)$  为高斯点的充分必要条件是: 在区间  $[-1, 1]$  内, 以这些点为零点的  $n+1$  次多项式  $\omega_{n+1}(x)$  与所有的次数都不超过  $n$  的多项式  $p(x)$  都正交, 即

$$\int_{-1}^1 p(x)\omega_{n+1}(x)dx = 0 \quad (8.5.5)$$

**证明** 首先证明必要性, 设  $p(x)$  是任意次数都不超过  $n$  的多项式, 则  $p(x)\omega_{n+1}(x)$  的次数不超过  $2n+1$ , 因此, 如果  $x_0, x_1, \dots, x_n$  是高斯点, 则求积公式 (8.5.1) 对于  $p(x)\omega(x)$  能准确成立, 即有

$$\int_{-1}^1 p(x)\omega_{n+1}(x)dx = \sum_{i=0}^n A_i p(x_i)\omega_{n+1}(x_i)$$

但  $\omega_{n+1}(x_i) = 0 (i = 0, 1, 2, \dots, n)$ , 故式 (8.5.5) 成立。

再证明充分性, 对于任意给定的次数不超过  $2n+1$  的多项式  $f(x)$ , 用  $\omega_{n+1}(x)$  除  $f(x)$ , 记商为  $p(x)$ , 余式为  $q(x)$ , 则  $p(x)$  和  $q(x)$  都是次数不超过  $n$  次的多项式, 且

$$f(x) = p(x)\omega_{n+1}(x) + q(x)$$

由式 (8.5.5) 得

$$\int_{-1}^1 f(x)dx = \int_{-1}^1 q(x)dx \quad (8.5.6)$$

由于所给求积公式 (8.5.1) 是插值型的, 它对于  $q(x)$  能准确成立, 因此

$$\int_{-1}^1 q(x)dx = \sum_{i=0}^n A_i q(x_i)$$

再注意, 由  $\omega(x_i) = 0 (i = 0, 1, \dots, n)$  可知  $q(x_i) = f(x_i) (i = 0, 1, \dots, n)$ , 从而有

$$\int_{-1}^1 q(x)dx = \sum_{i=0}^n A_i f(x_i)$$

于是由式 (8.5.6) 得

$$\int_a^b f(x)dx = \sum_{i=0}^n A_i f(x_i)$$

这说明式 (8.5.1) 对于一切次数不超过  $2n+1$  的多项式均能成立, 因此  $x_i (i = 0, 1, \dots, n)$  是高斯点。证毕。

## 8.5.2 高斯-勒让德求积公式

由定理 8.5.2 可知, 若能找到满足式 (8.5.5) 的  $n+1$  次多项式  $\omega_{n+1}(x)$ , 则求积公式的高斯点就能确定了, 从而确定了一个高斯求积公式。为解决这一问题, 这里介绍勒让德多项式 (参见 7.2.3 节) 及其相关结论。

**定理 8.5.3** 以区间  $[-1, 1]$  内的高斯点  $x_i (i = 0, 1, \dots, n)$  为零点的  $n+1$  次多项式为勒让德 (Legendre) 多项式。

**证明** 设  $p_n(x)$  为零点是高斯点的  $n$  次多项式, 考虑  $n$  重积分

$$u(x) = \underbrace{\int_{-1}^x \int_{-1}^x \cdots \int_{-1}^x}_{n} p_n(x) \underbrace{dx dx \cdots dx}_n$$

它是  $2n$  次多项式, 且  $u(x)$  的  $n$  阶导数为

$$u^{(n)}(x) = p_n(x) \quad (8.5.7)$$

同时还有

$$u(-1) = u'(-1) = \cdots = u^{(n-1)}(-1) = 0 \quad (8.5.8)$$

再设  $v(x)$  为任意  $n-1$  次多项式, 注意  $v^{(n)}(x) = 0$ , 利用式 (8.5.7) 和式 (8.5.8), 分步积分得

$$\begin{aligned}\int_{-1}^1 v(x) p_n(x) dx &= \int_{-1}^1 v(x) u^{(n)}(x) dx \\ &= v(1)u^{(n-1)}(1) - v'(1)u^{(n-2)}(1) + \cdots + (-1)^{n-1}v^{(n-1)}(1)u(1)\end{aligned}\quad (8.5.9)$$

另一方面, 由于  $p_n(x)$  的零点总是高斯点, 由定理 8.5.2 可知, 式 (8.5.9) 为 0, 再考虑到  $v(x)$  的任意性, 得

$$u(1) = u'(1) = \cdots = u^{(n-1)}(1) = 0 \quad (8.5.10)$$

式 (8.5.8) 和式 (8.5.10) 说明,  $-1$  和  $1$  都是  $u(x)$  的  $n-1$  重零点, 因而可设

$$u(x) = c(x^2 - 1)^n$$

式中,  $c$  为待定系数, 为使  $p_n(x)$  的首项系数为 1, 取  $c = \frac{n!}{(2n)!}$ 。

根据式 (8.5.7) 于是有

$$p_n(x) = \frac{n!}{(2n)!} \frac{d^n}{dx^n} [(x^2 - 1)^n] \quad (8.5.11)$$

证毕。

由式 (8.5.11) 可逐步构造出首项系数为 1 的勒让德多项式

$$\begin{aligned}p_1(x) &= x \\ p_2(x) &= x^2 - \frac{1}{3} \\ p_3(x) &= x^3 - \frac{3}{5}x \\ p_4(x) &= x^4 - \frac{30}{35}x^2 + \frac{3}{35} \\ &\dots\end{aligned}$$

根据定理 8.5.3, 利用勒让德多项式, 取它的零点作为求积节点, 即可构造出高斯公式, 称为高斯-勒让德求积公式。

当取一个节点  $x_0$  时, 因为  $p_1(x) = x$ , 其零点为  $x_0 = 0$ , 以  $x_0$  为节点, 其公式为

$$\int_{-1}^1 f(x) dx = A_0 f(0)$$

令此公式对  $f(x) = 1$  精确成立, 得  $A_0 = 2$ , 所以, 一个点的高斯-勒让德求积公式为

$$\int_{-1}^1 f(x) dx = 2f(0) \quad (8.5.12)$$

当取两个节点  $x_0, x_1$  时, 因为  $p_2(x) = x^2 - \frac{1}{3}$ , 其零点为

$$x_0 = -\frac{1}{\sqrt{3}}, \quad x_1 = \frac{1}{\sqrt{3}}$$

以  $x_0, x_1$  为节点构造其公式为

$$\int_{-1}^1 f(x) dx = A_0 f\left(-\frac{1}{\sqrt{3}}\right) + A_1 f\left(\frac{1}{\sqrt{3}}\right)$$

令此公式对  $f(x) = 1, x$  都精确成立, 得  $A_0 = A_1 = 1$ , 从而得两个点的高斯-勒让德求积公式

$$\int_{-1}^1 f(x) dx = f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right) \quad (8.5.13)$$

类似地, 可以求出三个点的高斯-勒让德求积公式

$$\int_{-1}^1 f(x) dx = \frac{5}{9} f\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9} f(0) + \frac{5}{9} f\left(\sqrt{\frac{3}{5}}\right) \quad (8.5.14)$$

当然，也可以利用代数精度方法设计高斯求积公式。

① 一点高斯求积公式

$$\int_{-1}^1 f(x) dx \approx A_0 f(x_0)$$

令此公式对  $f=1, x$  都精确成立，可列出方程

$$\begin{cases} A_0 = 2 \\ A_0 x_0 = 0 \end{cases}$$

据此定出  $A_0 = 2, x_0 = 0$ ，故得到一点高斯求积公式

$$\int_{-1}^1 f(x) dx = 2f(0)$$

它与式 (8.5.12) 相同。

② 两点高斯求积公式

$$\int_{-1}^1 f(x) dx \approx A_0 f(x_0) + A_1 f(x_1)$$

由对称性原理，令  $A_0 = A_1, x_0 = -x_1$ ，则所要设计的求积公式具有以下形式

$$\int_{-1}^1 f(x) dx \approx A_0 [f(x_0) + f(-x_0)]$$

由于它对于  $f=x, x^3$  都精确成立，故对称性原理正确，再令它对于  $f=1, x^2$  都精确成立，可列出方程组

$$\begin{cases} 2A_0 = 2 \\ 2A_0 x_0^2 = \frac{2}{3} \end{cases}$$

可得  $A_0 = 1, x_0 = -\frac{1}{\sqrt{3}}$ ，因而得到两点高斯公式

$$\int_{-1}^1 f(x) dx = f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right)$$

它与式 (8.5.13) 相同。

③ 三点高斯求积公式

$$\int_{-1}^1 f(x) dx \approx A_0 f(x_0) + A_1 f(x_1) + A_2 f(x_2)$$

由对称性原理令  $A_0 = A_2, x_0 = -x_2, x_1 = 0$ ，则所要设计的求积公式具有以下形式

$$\int_{-1}^1 f(x) dx \approx A_0 [f(x_0) + f(x_1)] + A_1 f(0)$$

由于它对于  $f=x, x^3, x^5$  精确成立，故对称性原理是正确的，再令它对  $f=1, x^2, x^4$  准确成立，可列出方程组

$$\begin{cases} A_0 + A_1 = 2 \\ 2A_0 x_0^2 = \frac{2}{3} \\ 2A_0 x_0^4 = \frac{2}{5} \end{cases}$$

将其中第 2 式与第 3 式相除，可得

$$x_0 = -\sqrt{\frac{3}{5}}, A_0 = \frac{5}{9}, A_1 = \frac{8}{9}$$

因而有三点高斯公式

$$\int_{-1}^1 f(x)dx = \frac{5}{9}f\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9}f(0) + \frac{5}{9}f\left(\sqrt{\frac{3}{5}}\right)$$

它与式(8.5.14)相同。

为了便于应用,可将高斯-勒让德求积公式中的节点和系数制成表,只要查表就可方便地写出相应的高斯-勒让德求积公式。表 8.5.1 给出了 1~4 个节点时的值。

表 8.5.1 节点和系数

$n$	节点 $x_i$	系数 $A_i$
0	0.000 000 0	2.000 000 0
1	$\pm 0.577\ 350\ 3$	1.000 000 0
2	$\pm 0.774\ 596\ 7$	0.555 555 56
	0.000 000 0	0.888 888 89
3	$\pm 0.861\ 136\ 3$	0.347 854 8
	$\pm 0.339\ 981\ 0$	0.652 145 2

例 8.5.1 用三个节点的高斯-勒让德求积公式求

$$I = \int_0^1 \frac{\sin x}{x} dx$$

解 因积分区间是 [0,1], 先做变换

$$x = \frac{1}{2}(t+1)$$

把区间 [0,1] 化为 [-1,1] 内的积分, 有

$$\int_0^1 \frac{\sin x}{x} dx = \int_{-1}^1 \frac{\sin \frac{1}{2}(t+1)}{t+1} dt$$

于是有

$$\begin{aligned} I &\approx 0.5555556 \times \frac{\sin \frac{1}{2}(-0.7745967+1)}{-0.7745967+1} + 0.8888889 \times \frac{\sin \frac{1}{2}}{0+1} + \\ &\quad 0.5555556 \times \frac{\sin \frac{1}{2}(0.7745967+1)}{0.7745967+1} = 0.9460831 \end{aligned}$$

与例 8.4.1 和例 8.4.2 的求解结果相对比, 例 8.4.1 用复合梯形求积公式计算, 对积分区间 [0,1] 二分 11 次, 用 2049 个函数值, 才取得 7 位有效数字; 例 8.4.2 用龙贝格积分公式, 对积分区间 [0,1] 二分 3 次, 用 9 个函数值, 取得了同样的结果。本例仅用 3 个函数值, 取得同样结果, 这说明高斯求积公式的精度是非常高的。

### 8.5.3 高斯求积公式的稳定性和收敛性

因为高斯点往往是无理数, 函数值  $f(x_i)$  ( $i=0,1,\dots,n$ ) 必然有舍入误差, 所以高斯求积公式的稳定性引人关注。事实上, 高斯求积公式不但精度高, 而且是数值稳定的。这主要是因为其

求积系数是正的。

**定理 8.5.4** 高斯求积公式 (8.5.1) 中的求积系数全是正的, 且有

$$A_i = \int_{-1}^1 [l_i(x)]^2 dx \quad (i = 0, 1, 2, \dots, n)$$

式中  $l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j}$ 。

**证明** 因为  $l_i(x)$  为  $n$  次多项式, 所以  $l_i^2(x)$  是  $2n$  次多项式, 由于式 (8.5.1) 具有  $2n+1$  次代数精度, 故  $2n$  次多项式  $l_i^2(x)$  能精确成立, 即有

$$\int_a^b l_i^2(x) dx = \sum_{j=0}^n A_j l_i^2(x_j) \quad (8.5.15)$$

注意

$$l_i(x_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

所以式 (8.5.15) 右端为  $A_i$ , 从而有

$$A_i = \int_{-1}^1 l_i^2(x) dx > 0$$

即式 (8.5.1) 的求积系数全为正。证毕。

对于高斯求积公式 (8.5.1), 若计算每个函数值  $f(x_i)$  时产生的舍入误差  $\varepsilon_i$ , 则其整个积分计算中产生的舍入误差为

$$\varepsilon = \sum_{i=0}^n A_i \varepsilon_i$$

从而有估计式

$$|\varepsilon| \leq \left( \sum_{i=0}^n |A_i| \right) \max_{0 \leq i \leq n} |\varepsilon_i|$$

若令式 (8.5.1) 中的  $f(x) = 1$ , 则有

$$\sum_{i=0}^n A_i = \int_{-1}^1 dx = 2$$

再由定理 8.5.3 可知,  $A_i > 0$ , 所以

$$|\varepsilon| \leq 2 \max_{0 \leq i \leq n} |\varepsilon_i|$$

这说明高斯求积公式的误差是可控制的, 从而具有较好的稳定性。

## 8.6 数值微分

### 8.6.1 中点公式

按照导数的定义, 导数  $f'(a)$  是差商  $\frac{f(a+h)-f(a)}{h}$  当  $h \rightarrow 0$  时的极限, 若要求精度不高, 则可将差商作为导数的近似值, 如此可得以下公式。

向前差商公式



$$f'(a) \approx \frac{f(a+h) - f(a)}{h} \quad (8.6.1)$$

向后差商公式

$$f'(a) \approx \frac{f(a) - f(a-h)}{h} \quad (8.6.2)$$

中心差商公式

$$f'(a) \approx \frac{f(a+h) - f(a-h)}{2h} = G(h) \quad (8.6.3)$$

中心差商公式 (8.6.3) 称为中点方法, 它是式 (8.6.1) 和式 (8.6.2) 的算术平均值, 其几何意义如图 8.6.1 所示。

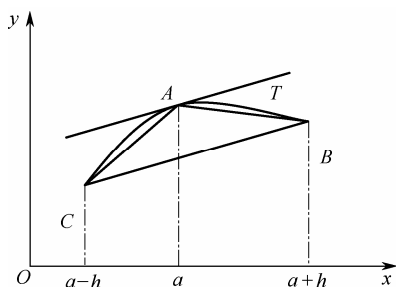


图 8.6.1 中点方法的几何意义

由图 8.6.1 可以看出, 3 种导数的近似值分别表示弦线  $AB$ ,  $AC$  和  $BC$  的斜率, 而导数值表示切线  $AT$  的斜率, 其中以  $BC$  的斜率更接近切线  $AT$  的斜率, 因此, 通常取中点公式来求导。

为了对中心差商公式 (8.6.3) 进行误差分析, 将  $f(a \pm h)$  在  $x=a$  处泰勒展开, 有

$$f(a \pm h) = f(a) \pm hf'(a) + \frac{h^2}{2!} f''(a) \pm \frac{h^3}{3!} f'''(a) + \frac{h^4}{4!} f^{(4)}(a) \pm \frac{h^5}{5!} f^{(5)}(a) + \dots \quad (8.6.4)$$

代入式 (8.6.3) 得

$$G(h) = \frac{f(a+h) - f(a-h)}{2h} = f'(a) + \frac{h^2}{3!} f'''(a) + \frac{h^4}{5!} f^{(5)}(a) + \dots = f'(a) + O(h^3) \quad (8.6.5)$$

由式 (8.6.5) 可以看出, 如果从截断误差的角度来看, 步长越小, 计算结果就越准确; 但如果从舍入误差的角度来看, 当  $h$  很小时, 由于  $f(a+h)$  与  $f(a-h)$  很接近, 直接相减会造成有效数字的严重丢失。在实际计算时, 通常在变步长的过程中实现步长的自动选择。

**例 8.6.1** 用变步长的中点方法求  $e^x$  在  $x=1$  处的导数值, 步长从  $h=0.8$  算起。

**解** 计算公式为

$$G(h) = \frac{e^{1+h} - e^{1-h}}{2h}$$

式中, 步长  $h = \frac{0.8}{2^k}$  ( $k$  为二分次数), 计算结果见表 8.6.1。

二分 9 次得到的结果  $f'(1) \approx 2.71828$ , 有 6 位有效数字。

为了提高中点方法的收敛速度, 可以利用外推算法的思想对式 (8.6.3) 进行加速。将式 (8.6.5) 的步长折半, 得

$$G\left(\frac{h}{2}\right) = f'(a) + \frac{1}{3!}\left(\frac{h^2}{2}\right)f'''(a) + \frac{1}{5!}\left(\frac{h}{2}\right)^4 f^{(5)}(a) + \cdots \quad (8.6.6)$$

表 8.6.1 例 8.6.1 的计算结果

$k$	$G(h)$	$k$	$G(h)$
0	3.017 65	...	...
1	2.791 35	9	2.718 28
2	2.736 44	10	2.718 28
3	2.722 81		

令

$$G_1(h) = \frac{4}{3}G\left(\frac{h}{2}\right) - \frac{1}{3}G(h)$$

利用式 (8.6.5) 和式 (8.6.6) 得

$$G_1(h) = f'(a) - \frac{1}{4 \times 5!}h^4 f^{(5)}(a) + \cdots = f'(a) + O(h^4) \quad (8.6.7)$$

同理, 令

$$G_2(h) = \frac{16}{15}G_1\left(\frac{h}{2}\right) - \frac{1}{15}G_1(h) = f'(a) + O(h^6) \quad (8.6.8)$$

令

$$G_3(h) = \frac{64}{63}G_2\left(\frac{h}{2}\right) - \frac{1}{63}G_2(h) \quad (8.6.9)$$

可得

$$G_3(h) = f'(a) + O(h^8)$$

这种加速过程还可以继续下去, 但效果已不再明显。

**例 8.6.2** 利用加速公式 (8.6.7)、式 (8.6.8) 和式 (8.6.9) 计算例 8.6.1 的值。

**解** 计算结果见表 8.6.2。

表 8.6.2 例 8.6.2 的计算结果

$h$	$G(h)$	$G_1(h)$	$G_2(h)$	$G_3(h)$
0.8	3.017 65			
0.4	2.791 35	2.715 917		
0.2	2.736 44	2.718 137	2.718 285	
0.1	2.722 81	2.719 267	2.718 276	2.718 28

计算式 (8.6.7)、式 (8.6.8) 和式 (8.6.9) 只需要简单的算术运算, 与求函数值  $f(a \pm h)$  相比可以忽略不计, 因此, 在相同计算量的情况下, 加速效果相当明显。

## 8.6.2 插值型微分公式

假设已知  $f(x)$  在节点  $x_i (i=0,1,2,\cdots,n)$  上的函数值为  $f(x_i)$ , 则可构建  $n$  次插值多项式  $L_n(x)$ , 并取  $L'_n(x)$  的值作为  $f'(x)$  的近似计算值, 即

$$f'(x) \approx L'_n(x) \quad (8.6.10)$$

式 (8.6.10) 称为插值型求积公式, 由插值余项定理可得, 式 (8.6.10) 的余项为

$$R'_n(x) = f'(x) - L'_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega'_{n+1}(x) + \frac{\omega_{n+1}(x)}{(n+1)!} \frac{d}{dx} f^{(n+1)}(\xi)$$

式中  $\omega_{n+1}(x) = \prod_{j=0}^n (x - x_j)$ 。

由于  $\xi$  是与  $x$  有关的未知函数, 对任意给定的点  $x$ , 上式第 2 项无法求出。但是, 如果只求其某个节点  $x_i$  上的函数值, 则上式第 2 项因  $\omega_{n+1}(x_i) = 0$  而等于 0, 这时, 有余项公式

$$R'_n(x_i) = f'(x_i) - L'_n(x_i) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega'_{n+1}(x_i) \quad (8.6.11)$$

设已给出三个节点  $x_0, x_1 = x_0 + h, x_2 = x_0 + 2h$  上的函数值, 可构建 2 次拉格朗日插值多项式

$$L_2(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f(x_0) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f(x_1) + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f(x_2)$$

$$R_2(x) = \frac{f'''(\xi)}{3!} (x-x_0)(x-x_1)(x-x_2)$$

所以

$$L'_2(x_i) = \frac{f(x_0)}{2h^2} (2x_i - x_1 - x_2) + \frac{f(x_1)}{-h^2} (2x_i - x_0 - x_2) + \frac{f(x_2)}{2h^2} (2x_i - x_0 - x_1)$$

$$R'_2(x_i) = \frac{f'''(\xi)}{3!} [(x_i - x_0)(x_i - x_1) + (x_i - x_0)(x_i - x_2) + (x_i - x_1)(x_i - x_2)]$$

$$f'(x_i) = L'_2(x_i) + R'_2(x_i)$$

分别取  $i=0,1,2$ , 可得带余项的三点求导式

$$\begin{cases} f'(x_0) = \frac{1}{2h} [-3f(x_0) + 4f(x_1) - f(x_2)] + \frac{h^2}{3} f'''(\xi) \\ f'(x_1) = \frac{1}{2h} [-f(x_0) + f(x_2)] - \frac{h^2}{6} f'''(\xi) \\ f'(x_2) = \frac{1}{2h} [f(x_0) - 4f(x_1) + 3f(x_2)] + \frac{h^2}{3} f'''(\xi) \end{cases} \quad \xi \in [x_0, x_2] \quad (8.6.12)$$

在式 (8.6.12) 中的第 2 式, 由于少用了一个函数值  $f(x)$ , 且精度比其他两个公式高一倍, 而引人关注。

但必须指出, 当插值多项式  $L_n(x)$  收敛于  $f(x)$  时, 不能保证  $L'_n(x)$  一定收敛于  $f'(x)$ , 而且当节点间的距离缩小时, 虽然截断误差缩小了, 但舍入误差却可能增大, 因此, 缩小步长不一定能提高计算结果的精度。

**例 8.6.3** 已知函数  $f(x) = e^x$  的函数值 (见表 8.6.3)。

表 8.6.3 例 8.6.3 的函数值

$x$	0	0.90	0.99	1.00	1.01	1.10	2
$f(x)=e^x$	1.000	2.460	2.691	2.718	2.746	3.004	7.389

试用式 (8.6.12) 的第 2 式计算  $f'(1)$  的近似值, 并比较步长。  $h$  分别取 1, 0.1, 0.01。

**解** (1)  $h=1$  时, 取  $x_0=0, x_1=1, x_2=2$

$$f'(1) \approx \frac{1}{2}(-e^0 + e^2) = 3.195$$

(2)  $h = 0.1$  时, 取  $x_0 = 0.90, x_1 = 1.00, x_2 = 1.10$

$$f'(1) \approx \frac{1}{2 \times 0.1}(-e^{0.90} + e^{1.10}) = 2.720$$

(3)  $h = 0.01$  时, 取  $x_0 = 0.99, x_1 = 1.00, x_2 = 1.01$

$$f'(1) = \frac{1}{2 \times 0.01}(-e^{0.99} + e^{1.01}) = 2.750 \quad (\text{中间结果取 3 位有效数字})$$

而  $f'(1)$  的真值为  $e^1 = 2.7182818$ 。

上式的计算结果表明, 当步长由 1 减小到 0.1 时, 计算精度明显提高, 但是当步长由 0.1 减小到 0.01 时, 精度反而有所下降。问题的根源在于实际计算时, 不但有截断误差的存在, 而且还有舍入误差的存在, 而数值微分恰好对舍入误差非常敏感, 它随  $h$  的逐渐缩小而误差逐渐增大, 这就是计算的不稳定性, 所以, 在计算数值微分时, 一定要进行误差分析。

## 本章小结

本章介绍数值积分和数值微分的基本思想, 先设法构造某一个简单函数  $p(x)$  近似表示  $f(x)$ , 然后对  $p(x)$  求积或求导, 从而推导出求  $f(x)$  的数值积分和数值微分公式。

插值型求积公式有两类: 一类是牛顿-柯特斯公式, 它是基于等距节点, 求积系数容易求得, 算法简单且容易编程, 但由于收敛性和稳定性都没有保证, 所以常用的是低阶复合公式。另一类是高斯求积公式, 它不但具有最高的代数精度, 而且收敛性和稳定性都有保证, 因此是高精度的求积公式, 高斯公式的主要缺点是节点与系数无规律, 所以不便编程实现, 在实际应用中, 可以把低阶高斯公式进行复合。

龙贝格算法是在积分区间逐次二分的过程中, 通过对梯形公式得到的近似值进行外推加速处理, 从而获得高精度的积分值, 由于可以自动选取步长, 便于编程实现。

数值微分的中点公式和插值型公式, 应用的难点在于步长  $h$  的选取,  $h$  过大, 截断误差变大,  $h$  过小, 舍入误差变大, 因此, 在实际计算时, 恰当地选取步长  $h$  是关键。

## 习题 8

8.1 确定下列求积公式中的待定参数, 使其代数精度尽量高, 并指明求积公式所具有的代数精度。

$$(1) \int_0^1 f(x) dx \approx Af(0) + Bf(x_1) + Cf(1)$$

$$(2) \int_{-2h}^{2h} f(x) dx \approx A_{-1}f(-h) + A_0f(0) + A_1f(h)$$

$$(3) \int_{-h}^h f(x) dx \approx Af(-h) + Bf(x_1)$$

8.2 确定下列求积公式中的待定参数, 使其代数精度尽量高, 并指明所构造出的求积公式所具有的代数精度。

$$(1) \int_{-h}^h f(x)dx \approx A_{-1}f(-h) + A_0f(0) + A_1f(h)$$

$$(2) \int_{-1}^1 f(x)dx \approx \frac{f(-1) + 2f(x_1) + 3f(x_2)}{3}$$

$$(3) \int_0^h f(x)dx \approx \frac{h[f(0) + f(h)]}{2} + \alpha h^2[f'(0) - f'(h)]$$

8.3 证明求积公式  $\int_{x_0}^{x_1} f(x)dx \approx \frac{h}{2}(f(x_0) + f(x_1)) - \frac{h^2}{12}(f'(x_1) - f'(x_0))$  具有 3 次代数精度，式中， $h = x_1 - x_0$ 。

8.4 对求积公式  $\int_0^1 f(x)dx \approx A_0f(0) + A_1f(1) + B_0f'(0)$ ，已知其余项表式为  $R(f) = kf'''(\varepsilon)$ ， $\varepsilon \in (0,1)$ 。试确定系数  $A_0, A_1$  及  $B_0$ ，使该求积公式具有尽可能高的代数精度，并给出代数精度的次数及求积公式余项。

8.5 推导下列 3 种矩形求积公式

$$\int_a^b f(x)dx = (b-a)f(a) + \frac{1}{2}f'(\eta)(b-a)^2, \quad \eta \in (a,b) \quad (\text{左矩形})$$

$$\int_a^b f(x)dx = (b-a)f(b) - \frac{1}{2}f'(\eta)(b-a)^2, \quad \eta \in (a,b) \quad (\text{右矩形})$$

$$\int_a^b f(x)dx = (b-a)f\left(\frac{a+b}{2}\right) + \frac{1}{24}f''(\eta)(b-a)^3, \quad \eta \in (a,b) \quad (\text{中矩形})$$

8.6 设  $I = \int_0^1 f(x)dx \approx A_0f\left(\frac{1}{4}\right) + A_1f\left(\frac{1}{2}\right) + A_2f\left(\frac{3}{4}\right)$  是插值型的，试确定参数  $A_0, A_1, A_2$ ，并指出此求积公式具有的代数精度。

8.7 建立高斯求积公式  $\int \frac{1}{\sqrt{x}} f(x)dx \approx A_0f(x_0) + A_1f(x_1)$ 。

8.8 试确定常数  $A, B, C$  及  $a$ ，使求积公式  $\int_{-2}^2 f(x)dx \approx Af(-a) + Bf(0) + Cf(a)$  有尽可能高的代数精度，并指出所得求积公式的代数精度，它是否为高斯求积公式？

8.9 确定如下求积公式及截断误差表示式

$$(1) \int_{-1}^1 f(x)dx \approx A_1f\left(-\frac{1}{2}\right) + A_2f(0) + A_3f\left(\frac{1}{2}\right)$$

$$(2) \int_0^h f(x)dx \approx A_0f(0) + B_0f'(0) + A_1f(h) + B_1f'(h)$$

$$(3) \int_0^{2h} f(x)dx \approx A_0f(0) + A_1f(h) + A_2f(2h)$$

$$(4) \int_{-1}^1 x^2 f(x)dx \approx A_0f(x_0) \text{ 或 } A_1f(x_1) + A_2f(x_2)$$

$$(5) \int_0^1 \sqrt{x} f(x)dx \approx A_0f(x_0) \text{ 或 } A_1f(x_1) + A_2f(x_2)$$

8.10 试构造下列求积公式，使其代数精度尽量高，并证明所构造出的求积公式是插值型的

$$\int_0^1 f(x)dx \approx A_0f\left(\frac{1}{4}\right) + A_1f\left(\frac{3}{4}\right)$$

8.11 判别下列求积公式是否为插值型的，并指明其代数精度

$$\int_0^3 f(x)dx \approx \frac{3}{2}[f(1)+f(2)]$$

8.12 试设计求积公式

$$\int_0^1 f(x)dx \approx A_0 f(0) + A_1 f(1) + B_0 f'(0)$$

8.13 试设计求积公式

$$\int_0^h f(x)dx \approx h[a_0 f(0) + a_1 f(1)] + h^2[b_0 f'(0) + b_1 f'(1)]$$

8.14 试设计求积公式

$$\int_a^b f(x)dx \approx A_0 f(a) + A_1 f\left(\frac{a+b}{2}\right) + A_2 f(b) + B_0 f'(a) + B_1 f'\left(\frac{a+b}{2}\right) + B_2 f'(b)$$

8.15 试设计求积公式

$$\int_{-1}^1 f(x)dx \approx A[f(x_0) + f(x_1) + f(x_2)], x_0 < x_1 < x_2$$

8.16 证明下列求积公式有 5 阶代数精度, 并说明它是高斯三点公式

$$\int_1^3 f(x)dx \approx \frac{5}{9}f(2-\sqrt{\frac{3}{5}}) + \frac{8}{9}f(2) + \frac{5}{9}f(2+\sqrt{\frac{3}{5}})$$

8.17 试设计求积公式

$$\int_{-2}^2 f(x)dx \approx Af(-a) + Bf(0) + Cf(a)$$

8.18 试设计下列带权的高斯公式

$$\int_0^1 \sqrt{x} f(x)dx \approx A_0 f(x_0) + A_1 f(x_1)$$

8.19 下列求积公式称为辛普生 3/8 公式

$$\int_0^3 f(x)dx \approx \frac{3}{8}[f(0) + 3f(1) + 3f(2) + f(3)]$$

试判断这个公式的代数精度。

8.20 用三点高斯公式求下列积分值  $\pi = \int_0^1 \frac{4}{1+x^2} dx$ 。

8.21 设  $f(x) = x^3$ , 对  $h = 0.1$  和  $h = 0.01$ , 用中心差商公式计算  $f'(2)$  的近似值。

8.22 用三点公式和积分方法求  $f(x) = \frac{1}{(1+x)^2}$  在  $x=1.0, 1.1$  和  $1.2$  处的导数值, 并估计误差。

$f(x)$  的值如下:

$x$	1.0	1.1	1.2
$f(x)$	0.2500	0.2268	0.2066

8.23 (1) 5 个节点的牛顿-柯特斯求积公式的代数精度为多少? 5 个节点的求积公式最高代数精度为多少?

(2) 要使求积公式  $\int_0^1 f(x)dx \approx \frac{1}{4}f(0) + A_1 f(x_1)$  具有 2 次代数精度, 求  $x_1$  和  $A_1$ 。

(3) 若用复化梯形公式计算积分  $\int_0^{1.5} e^{-x} dx$ , 将  $[0, 1.5]$  区间  $n$  等分,  $n$  取什么值才能使截断

误差不超过  $\frac{1}{2} \times 10^{-4}$ ?

(4) 若  $f(x) = \sqrt{x+2}$ , 取  $h=0.5$ , 用中心差商求导公式, 求  $f'(0.5)$ 。

8.24  $n+1$  点插值型数值积分公式  $\int_a^b f(x)dx \approx \sum_{n=0}^n A_n f(x_n)$  的代数精度至少为多少? 最高不超过多少?

8.25 求积公式  $\int_0^1 f(x)dx \approx \frac{3}{4}f(\frac{1}{3}) + \frac{1}{4}f(1)$  的代数精度为多少?  $\int_0^1 f(x)dx \approx \frac{3}{4}f(\frac{1}{3}) + \frac{1}{4}f(1)$  的余项表达式为  $kf'''(\varepsilon)$ , 式中  $\varepsilon \in (0,1)$ , 求  $k = ?$

## 第9章 常微分方程初值问题的数值解法



### 学习要点

常微分方程的数值解法是, 利用数值微分、数值积分和泰勒展开等离散化方法将微分方程变为差分方程进行计算, 本章主要内容有:

- (1) 欧拉(Euler)公式, 包括显式、隐式、两步、改进的欧拉公式和梯形公式。
- (2) 龙格-库塔方法, 包括二阶、四阶龙格-库塔方法。
- (3) 单步法的局部截断误差、收敛性、稳定性。



### 教学建议

要求掌握欧拉公式及其变形公式的构造, 并能正确应用这些公式求微分方程的数值解。理解龙格-库塔方法的基本思想, 了解二阶龙格-库塔方法的推导过程, 能用经典的四阶龙格-库塔方法求微分方程数值解, 了解单步法的收敛性和稳定性, 能推导常用单步法的稳定区域。建议学时 4~6 学时。

## 9.1 引言

在第2章中, 我们介绍了常微分方程的基本知识, 包括常微分方程初值问题及其解的存在唯一性条件等。然而, 在实际中, 除了少数特殊类型的常微分方程(如常系数线性的, 可分离的)能用初等积分法求得其精确解外, 在大多数情况下, 要解出常微分方程解的解析表达式是极其困难的, 甚至是不可能的。因此, 有必要研究常微分方程初值问题的数值解法。

所谓初值问题的数值解法, 就是在给定初始点  $x_0$  的函数值  $y_0$  后, 能计算出精确解  $y(x)$  在自变量  $x$  的一系列后续离散节点  $x_1 < x_2 < \cdots < x_{n-1} < x_n$  处的近似解  $y_1, y_2, \cdots, y_{n-1}, y_n$  的方法。

我们把  $y_k$  ( $k=1, 2, 3, \cdots, n$ ) 称为初值问题在点列  $x_k$  上的数值解。相邻两个节点间的距离称为步长。为便于计算, 一般都把  $h$  取为定步长, 此时  $x_n = x_0 + nh$ 。在具体计算时, 可以从初值条件  $y(x_0) = y_0$  出发, 先求出  $y_1$ , 再由已知信息  $y_0, y_1$  求出  $y_2$ , 依次递推直至求出  $y_n$  为止。这种按节点  $x_1, x_2, \cdots, x_n$  的次序逐步向前推进的求解方法称为“步进式”方法。如果计算  $y_n$  时, 只利用前一步的函数值  $y_{n-1}$ , 则称这种方法为单步法; 如果在计算  $y_n$  时, 不仅利用  $y_{n-1}$ , 而且还要利用  $y_{n-2}, y_{n-3}, \cdots, y_{n-r}$ , 则称这种方法为  $r$  步方法, 也称多步法。本章只介绍单步法。

用数值计算方法求解常微分方程的初值问题时, 常常对连续的初值问题进行离散化处理。下面介绍几种常用的离散化方法。

### 1. 基于数值微分的离散化方法

在常微分方程初值问题



$$\begin{cases} y'(x) = f(x, y) & (9.1.1) \\ y(x_0) = y_0 & (9.1.2) \end{cases}$$

中, 如果将点  $x_n$  处的导数  $y'(x_n)$  用点  $x_n$  处的差商近似代替

$$y'(x_n) \approx \frac{y(x_{n+1}) - y(x_n)}{h} \quad (n = 0, 1, 2, \dots) \quad (9.1.3)$$

即  $y(x_{n+1}) \approx y(x_n) + hy'(x_n)$ , 式中,  $x_n + h = x_{n+1}$ 。

又因为  $y'(x_n) = f(x_n, y(x_n))$ , 得

$$y(x_{n+1}) \approx y(x_n) + hf(x_n, y(x_n)) \quad (n = 1, 2, 3, \dots)$$

用近似值  $y_n$ ,  $y_{n+1}$  分别代替上式中的精确值  $y(x_n)$ ,  $y(x_{n+1})$ , 由此, 可构造出初值问题的离散化方程

$$y_{n+1} = y_n + hf(x_n, y_n) \quad (9.1.4)$$

式 (9.1.4) 称为求解一阶常微分方程初值问题的欧拉 (Euler) 公式, 也称显式欧拉公式。

## 2. 基于数值积分的离散化方法

将微分方程的初值问题式 (9.1.1) 两边在区间  $[x_n, x_{n+1}]$  内积分得

$$\int_{x_n}^{x_{n+1}} y'(x) dx = \int_{x_n}^{x_{n+1}} f(x, y) dx$$

即

$$y(x_{n+1}) - y(x_n) = \int_{x_n}^{x_{n+1}} f(x, y(x)) dx$$

再将右边积分利用数值积分公式计算其近似值, 例如利用左矩形公式计算右边积分的近似值得

$$y(x_{n+1}) \approx y(x_n) + hf(x_n, y(x_n))$$

用近似值  $y_n$ ,  $y_{n+1}$  分别代替上式中的精确值  $y(x_n)$ ,  $y(x_{n+1})$ , 可得

$$y_{n+1} = y_n + hf(x_n, y_n)$$

由此构造出与式 (9.1.4) 相同的欧拉公式。

## 3. 基于泰勒展开的离散化方法

设  $y(x)$  是微分方程  $y'(x) = f(x, y)$  的一个解, 且函数  $f(x, y)$  充分可微, 则可利用泰勒展开式将式 (9.1.1) 离散化。

设  $y(x_n + h)$  在点  $x_n$  处的泰勒展开式为

$$y(x_n + h) = y(x_n) + hy'(x_n) + \frac{h^2}{2!} y''(x_n) + \dots + \frac{h^p}{p!} y^{(p)}(x_n) + O(h^{p+1})$$

取上式的线性部分, 并注意  $x_n + h = x_{n+1}$ ,  $y'(x_n) = f(x_n, y(x_n))$ , 可得

$$y(x_{n+1}) \approx y(x_n) + hf(x_n, y(x_n))$$

同样, 用近似值  $y_n, y_{n+1}$  代替精确值  $y(x_n), y(x_{n+1})$ , 可得

$$y_{n+1} = y_n + hf(x_n, y_n)$$

由此, 同样可以构造出与式 (9.1.4) 相同的欧拉公式。

常微分方程的数值解法就是把连续的初值问题进行离散化的方法, 其特点是, 只要初值问题的右端函数  $f(x, y)$  是可计算的, 就能应用数值计算方法, 因此, 具有通用性。而且只要对函数  $f(x, y)$  进行多次计算, 就能够确保数值计算方法的误差充分小。

## 9.2 欧拉公式

### 9.2.1 欧拉公式及其意义

由 9.1 节的求解微分方程的离散化过程，可得欧拉（Euler）公式

$$y_{n+1} = y_n + hf(x_n, y_n)$$

利用欧拉公式求常微分方程数值解的方法称为欧拉方法。

欧拉公式的几何意义非常明显，因为微分方程 (9.1.1) 的解在  $XOY$  平面上表示为一族积分曲线。其中通过点  $p_0(x_0, y_0)$  的那条积分曲线  $y = y(x)$  为初值问题式 (9.1.1)，式 (9.1.2) 的解。用欧拉公式求数值解的几何意义是：先在初始点  $p_0(x_0, y_0)$  处作积分曲线  $y = y(x)$  的切线，切线的斜率为  $f(x_0, y_0)$ ，记此切线与直线  $x = x_1$  交点  $p_1$  的纵坐标为  $y_1$ ，然后过点  $p_1(x_1, y_1)$  以  $f(x_1, y_1)$  为斜率作一条直线，记它与直线  $x = x_2$  交点  $p_2$  的纵坐标为  $y_2$ ，……，如此继续下去，可得一条折线  $p_0p_1p_2 \cdots p_n$ ，如图 9.2.1 所示。容易验证，该折线各个顶点的纵坐标  $y_n (n=1, 2, \cdots)$  就是欧拉公式 (9.1.4) 计算得到的近似解，所以，欧拉方法又称为折线法。

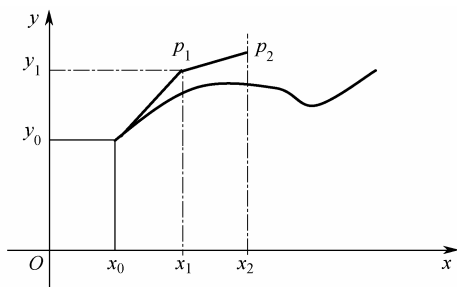


图 9.2.1 欧拉公式的几何意义

欧拉公式的算法框图如图 9.2.2 所示。

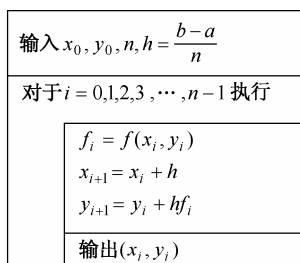


图 9.2.2 欧拉公式的算法框图

**例 9.2.1** 用欧拉方法求解初值问题

$$\begin{cases} y'(x) = y - x & 0 < x \leq 1, |y| < \infty \\ y(0) = 2 \end{cases}$$

步长  $h$  分别为 0.2, 0.1, 0.05, 0.01, 0.005。

**解** 用各种不同步长计算得到的离散节点  $x_1 = 0.2$ ,  $x_2 = 0.4$ ,  $x_3 = 0.6$ ,  $x_4 = 0.8$ ,  $x_5 = 1.0$  上的数值解及其精确解，见表 9.2.1。

表 9.2.1 例 9.2.1 的计算结果

$x_n$	$y_n$					精确解
	$h = 0.2$	$h = 0.1$	$h = 0.05$	$h = 0.01$	$h = 0.005$	$y(x) = e^x + x + 1$
0.0	2.0000	2.0000	2.0000	2.0000	2.0000	2.0000
0.2	2.4000	2.4100	2.4155	2.4202	2.4208	2.4214
0.4	2.8400	2.8641	2.8775	2.8889	2.8903	2.8918
0.6	3.3280	3.3716	3.3959	3.4167	3.4194	3.4221
0.8	3.8736	3.9436	3.9829	4.0167	4.0211	4.0255
1.0	4.4883	4.5937	4.6533	4.7084	4.7115	4.7183

从本例可以看出,  $h$  越小, 数值解  $y_i$  越精确。

## 9.2.2 欧拉公式的变形

在对微分方程初值问题进行离散化时, 如果用差商  $\frac{y(x_{n+1}) - y(x_n)}{h}$  代替方程  $y'(x_{n+1}) = f(x_{n+1}, y(x_{n+1}))$  中的  $y'(x_{n+1})$ , 并用近似值  $y_{n+1}$  表示  $y(x_{n+1})$ , 用近似值  $y_n$  表示  $y(x_n)$ , 可得

$$y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}) \quad (9.2.1)$$

式 (9.2.1) 称为隐式欧拉公式, 或后退的欧拉公式, 它是关于  $y_{n+1}$  的一个函数方程, 其计算远比显式欧拉公式 (9.1.4) 困难, 但式 (9.2.1) 的稳定性相对比较好。

如果将显式欧拉公式 (9.1.4) 和隐式欧拉公式 (9.2.1) 进行算术平均, 则得

$$y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y_{n+1})] \quad (9.2.2)$$

式 (9.2.2) 称为梯形公式, 它也是关于  $y_{n+1}$  的函数方程, 是隐式方法。

对用隐式方法解微分方程的初值问题, 如果右端函数  $f(x, y)$  是  $y$  的线性函数, 则隐式公式可显式计算, 如  $y'(x) = xy - 10$ , 其隐式欧拉公式为  $y_{n+1} = y_n + h(x_{n+1} \cdot y_{n+1} - 10)$ , 它有显式公式  $y_{n+1} = \frac{y_n - 10h}{1 - hx_{n+1}}$ 。但当  $f(x, y)$  是  $y$  的非线性函数时, 如  $y'(x) = \sqrt{y} - 10x$ , 其隐式公式  $y_{n+1} = y_n + h(\sqrt{y_{n+1}} - 10x_{n+1})$  为关于  $y_{n+1}$  的非线性方程, 此时可以用迭代法求解。以梯形公式为例, 用显式欧拉公式提供迭代初值  $y_{n+1}^{(0)}$ , 其迭代公式为

$$\begin{cases} y_{n+1}^{(0)} = y_n + hf(x_n, y_n) \\ y_{n+1}^{(k+1)} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y_{n+1}^{(k)})] \end{cases} \quad (k = 0, 1, 2, \dots) \quad (9.2.3)$$

反复迭代, 直到  $|y_{n+1}^{(k+1)} - y_{n+1}^{(k)}| < \varepsilon$ 。在迭代过程中, 步长  $h$  为迭代参数, 它需要满足一定的条件方能收敛, 将式 (9.2.2) 减去式 (9.2.3), 得

$$y_{n+1} - y_{n+1}^{(k)} = \frac{h}{2}[f(x_{n+1}, y_{n+1}) - f(x_{n+1}, y_{n+1}^{(k)})]$$

假设  $f(x, y)$  关于  $y$  满足 Lipschitz 条件, 即

$$|f(x, y_1) - f(x, y_2)| \leq L |y_1 - y_2| \quad (L \text{ 为 Lipschitz 常数})$$

则有

$$|y_{n+1} - y_{n+1}^{(k+1)}| \leq L \frac{h}{2} |y_{n+1} - y_{n+1}^{(k)}|$$

当  $L\frac{h}{2} < 1$ , 即  $h < \frac{2}{L}$  时, 由式 (9.2.3) 生成的迭代序列  $\{y_{n+1}^{(k)}\}$  收敛于  $y_{n+1}$ 。

对于隐式公式通常采用预测-校正技术, 即先用显式方法计算, 预测一个估计值  $\bar{y}_{n+1}$ , 为隐式公式提供一个好的迭代初值, 然后用隐式公式迭代一次, 得  $y_{n+1}$ 。例如, 用显式欧拉公式预测迭代初值, 梯形公式进行校正, 得

$$\begin{cases} \bar{y}_{n+1} = y_n + hf(x_n, y_n) \\ y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, \bar{y}_{n+1})] \end{cases} \quad (9.2.4)$$

上式也可写为

$$y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y_n + hf(x_n, y_n))] \quad (9.2.5)$$

式 (9.2.4) 和式 (9.2.5) 都称为改进的欧拉公式。为了编程方便, 常将改进的欧拉公式改写为

$$\begin{cases} T_1 = y_n + hf(x_n, y_n) \\ T_2 = y_n + hf(x_{n+1}, T_1) \\ y_{n+1} = \frac{T_1 + T_2}{2} \end{cases} \quad (9.2.6)$$

相应的算法框图如图 9.2.3 所示。

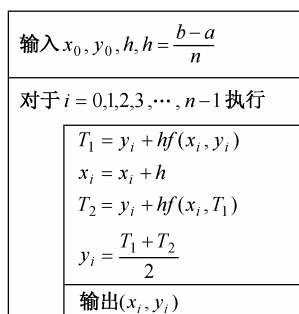


图 9.2.3 改进的欧拉公式算法框图

另外, 在对微分方程初值问题进行离散化时, 如果用中心差商  $\frac{y(x_{n+1}) - y(x_{n-1}))}{2h}$  代替方程  $y'(x_n) = f(x_n, y(x_n))$  中的  $y'(x_n)$ , 并用近似值  $y_{n-1}$  表示  $y(x_{n-1})$ , 用近似值  $y_{n+1}$  表示  $y(x_{n+1})$ , 可得

$$y_{n+1} = y_{n-1} + 2hf(x_n, y_n) \quad (9.2.7)$$

式 (9.2.7) 称为两步欧拉公式。前面介绍的 (显式) 欧拉公式、隐式欧拉公式、梯形公式和改进的欧拉公式, 它们都是单步法, 其特点是在计算  $y_{n+1}$  时只用到前一步的信息  $y_n$ , 而两步欧拉公式 (9.2.7) 除了用到  $y_n$  外, 还用到更前面的信息  $y_{n-1}$ , 因而属于多步法。

**例 9.2.2** 给定初值问题

$$\begin{cases} \frac{dy}{dx} = 2xy & 0 \leq x \leq 1 \\ y(0) = 1 \end{cases}$$

取  $h = 0.1$ , 分别用显式欧拉公式、隐式欧拉公式、梯形公式、改进的欧拉公式和两步欧拉公式计算其数值解并与精确值进行比较。

**解** 将原方程改写为  $\frac{1}{y} dy = 2x dx$  两边积分得

$$\ln y(x) = x^2 + c_1$$

所以  $y(x) = e^{x^2+c_1} = e^{c_1} \cdot e^{x^2} = ce^{x^2}$

式中， $c$  为任意常数。

由初值条件  $y(0)=1$ ，可得  $c=1$ 。

所以精确解为  $y(x)=e^{x^2}$ 。

因为  $f(x,y)=2xy$ ，所以各公式计算如下。

欧拉公式

$$y_{n+1} = y_n + h \cdot 2x_n y_n = (1 + 2hx_n) y_n$$

隐式欧拉公式

$$y_{n+1} = y_n + 2hx_{n+1}y_{n+1}$$

可以解出  $y_{n+1}$  得

$$y_{n+1} = \frac{y_n}{(1 - 2hx_{n+1})}$$

梯形公式

$$y_{n+1} = y_n + \frac{h}{2}(2x_n y_n + 2x_{n+1} y_{n+1})$$

整理得

$$y_{n+1} = \frac{(1 + hx_n)}{(1 - hx_{n+1})} y_n$$

两步欧拉公式

$$y_{n+1} = y_{n-1} + 4hx_n y_n$$

其前两步  $y_0, y_1$  的值分别由初值条件和欧拉公式给出。

改进的欧拉公式：

$$y_{n+1} = y_n + \frac{h}{2}[2x_n y_n + 2x_{n+1}(y_n + h \cdot 2x_n y_n)] = y_n + hy_n(x_n + x_{n+1} + 2x_n x_{n+1})$$

取  $n=0,1,2,\dots,10$ ，计算结果见表 9.2.2。

表 9.2.2 例 9.2.2 的计算结果

$i$	$x_i$	欧拉公式 $y_i$	隐式公式 $y_i$	梯形法 $y_i$	改进公式 $y_i$	两步公式 $y_i$	精确值 $y(x_i)$
0	0	1	1	1	1	1	1
1	0.1	1	1.020 408 2	1.010 101 0	1.01	1	1.010 050 167
2	0.2	1.02	1.062 925 2	1.041 022 5	1.040 704	1.04	1.040 810 774
3	0.3	1.060 8	1.130 771 5	1.094 683 5	1.093 988 0	1.083 2	1.094 174 284
4	0.4	1.124 448	1.229 099 5	1.174 504 1	1.173 192 8	1.169 984	1.173 510 871
5	0.5	1.214 403 84	1.365 666 1	1.285 772 9	1.283 472 9	1.270 397 44	1.284 025 417
6	0.6	1.335 844 224	1.551 893 3	1.436 235 7	1.432 355 8	1.424 063 45	1.433 329 415
7	0.7	1.496 145 531	1.804 527 1	1.636 999 9	1.630 593 8	1.612 172 68	1.632 316 22
8	0.8	1.705 605 905	2.148 246 5	1.903 902 0	1.893 445 5	1.875 471 84	1.896 480 879
9	0.9	1.978 502 84	2.619 812 8	2.259 576 0	2.242 596 9	2.212 323 67	2.247 907 987
10	1.0	2.334 633 35	3.274 760 0	2.736 597 6	2.709 057	2.671 908 36	2.718 281 828

由表 9.2.2 的计算结果可以看出, 梯形公式和改进的欧拉公式的计算结果误差比较小, 而欧拉公式和隐式欧拉公式的计算结果误差比较大。

### 9.3 单步法的局部截断误差和方法的阶

现在将解微分方程初值问题的单步法写成如下统一的形式:

$$y_{n+1} = y_n + h\varphi(x_n, x_{n+1}, y_n, y_{n+1}, h) \quad (9.3.1)$$

式中,  $\varphi$  与微分方程初值问题的右端函数  $f(x, y)$  有关, 称为增量函数。若  $\varphi$  中不含  $y_{n+1}$ , 则此方法是显式的, 否则是隐式的。

例如, 欧拉公式和改进的欧拉公式的增量函数  $\varphi$  分别为

$$f(x_n, y_n), \quad \frac{1}{2}[f(x_n, y_n) + f(x_{n+1}, x_n + hf(x_n, y_n))]$$

因此这两个公式都是显式公式。隐式欧拉公式和梯形公式的增量函数  $\varphi$  分别为

$$f(x_{n+1}, y_{n+1}), \quad \frac{1}{2}[f(x_n, y_n) + f(x_{n+1}, y_{n+1})]$$

它们都包含  $y_{n+1}$  项, 因此, 这两个公式都是隐式公式。

不论显式公式, 还是隐式公式, 从  $x_0$  开始计算, 如果考虑每步产生的误差, 直到  $x_n$ , 则有误差  $e_n = y(x_n) - y_n$ , 称为数值计算方法在  $x_n$  处的整体截断误差。一般地, 分析和求出整体截断误差  $e_n$  是非常困难的。为此, 我们仅考虑从  $x_n$  到  $x_{n+1}$  的局部情况。假设  $x_n$  处的值  $y_n$  没有误差, 即  $y_n = y(x_n)$ , 给出单步法的局部截断误差概念。

**定义 9.3.1 (局部截断误差)** 设  $y(x)$  是微分方程的精确解, 则

$$T_{n+1} = y(x_{n+1}) - y(x_n) - h\varphi(x_n, x_{n+1}, y(x_n), y(x_{n+1}), h) \quad (9.3.2)$$

称为单步法公式 (9.3.1) 的局部截断误差。

例如, 显式欧拉公式的局部截断误差为

$$T_{n+1} = y(x_{n+1}) - y(x_n) - hf(x_n, y(x_n)) \quad (9.3.3)$$

若用  $e_{n+1}$  表示欧拉公式在  $x_n$  处的整体截断误差, 则  $T_{n+1}$  与  $e_{n+1}$  之间的联系如图 9.3.1 所示。

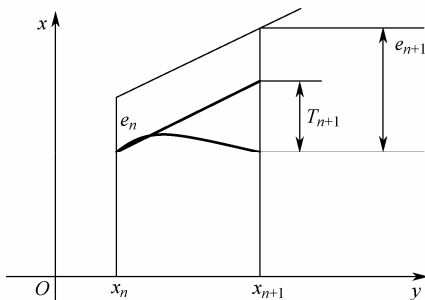


图 9.3.1  $T_{n+1}$  与  $e_{n+1}$  之间的联系

现在, 推导式 (9.3.3), 为此将  $y(x_{n+1})$  在  $x_n$  处泰勒展开, 并用  $y'$  代替  $f$  得

$$\begin{aligned} T_{n+1} &= y(x_{n+1}) - y(x_n) - hf(x_n, y(x_n)) \\ &= y(x_n) + hy'(x_n) + \frac{h^2}{2}y''(x_n) + O(h^3) - y(x_n) - hy'(x_n) \end{aligned}$$

$$\begin{aligned}
&= \frac{h^2}{2} y''(x_n) + O(h^3) \\
&= O(h^2)
\end{aligned}$$

另一方面, 当微分方程初值问题的解  $y(x)$  为一次多项式  $ax+b$  时, 在第  $n$  步精确, 即  $y_n = y(x_n)$  的前提下, 用欧拉公式可求第  $n+1$  步的数值解  $y_{n+1}$ 。

由于  $y(x) = ax+b$ ,  $y'(x) = a$ ,  $y_n = y(x_n)$  则

$$\begin{aligned}
y_{n+1} &= y_n + hf(x_n, y_n) \\
&= y(x_n) + hf(x_n, y(x_n)) \\
&= y(x_n) + hy'(x_n) \\
&= ax_n + b + ah \\
&= a(x_n + h) + b \\
&= ax_{n+1} + b \\
&= y(x_{n+1})
\end{aligned}$$

这说明, 当微分方程的解是一次多项式时, 欧拉公式的局部截断误差为 0, 我们称欧拉公式是一阶方法。对于一般公式, 有以下定义。

**定义 9.3.2 (方法的阶)** 如果求微分方程数值计算方法的局部截断误差是  $T_{n+1} = O(h^{p+1})$ , 式中  $p \geq 1$  为整数, 则称该方法是  $p$  阶的, 或该方法具有  $p$  阶精度。  $p$  越大, 方法的精度越高。含  $h^{p+1}$  的项, 称为该方法的局部截断误差主项。

由前面的推导可知, 欧拉公式是一阶方法, 其截断误差主项为  $\frac{h^2}{2} y''(x_n)$ 。

对于隐式欧拉公式, 有

$$\begin{aligned}
T_{n+1} &= y(x_{n+1}) - y(x_n) - hf(x_{n+1}, y(x_{n+1})) \\
&= y(x_{n+1}) - y(x_n) - hy'(x_{n+1}) \\
&= y(x_n) + hy'(x_n) + \frac{h^2}{2} y''(x_n) + O(h^3) - y(x_n) - h(y'(x_n) + hy''(x_n) + O(h^2)) \\
&= -\frac{h^2}{2} y''(x_n) + O(h^3) \\
&= O(h^2)
\end{aligned}$$

所以, 隐式欧拉公式也是一阶方法, 它的局部截断误差的主项是  $-\frac{h^2}{2} y''(x_n)$ 。

对于梯形公式, 有

$$\begin{aligned}
T_{n+1} &= y(x_{n+1}) - y(x_n) - \frac{h}{2} [f(x_n, y(x_n)) + f(x_{n+1}, y(x_{n+1}))] \\
&= y(x_n + h) - y(x_n) - \frac{h}{2} [y'(x_n) + y'(x_n + h)] \\
&= -\frac{h^3}{12} y'''(x_n) + O(h^4)
\end{aligned}$$

所以, 梯形公式是二阶方法, 其局部截断误差为  $-\frac{h^3}{12} y'''(x_n)$ 。

可以证明, 改进的欧拉方法也为二阶方法。

从例 9.2.2 的计算结果中也可以明显地看出, 梯形公式和改进的欧拉公式要比欧拉公式和隐式欧拉公式的精度高, 这也进一步验证了二阶方法比一阶方法好。

例 9.3.1 初值问题  $\begin{cases} y' = ax + b \ (x > 0) \\ y(0) = 0 \end{cases}$  有解  $y(x) = ax^2/2 + bx$ , 试证明改进的欧拉方法能准

确求解上述问题。

证明 记  $f(x, y) = ax + b, x_i = ih \ (i = 0, 1, 2, \dots, n)$ , 则改进的欧拉公式为

$$\begin{aligned} y_{i+1} &= y_i + \frac{h}{2} [f(x_i, y_i) + f(x_{i+1}, y_i + hf(x_i, y_i))] \\ &= y_i + \frac{h}{2} [(ax_i + b) + (ax_{i+1} + b)] \quad (i = 0, 1, 2, \dots, n-1) \end{aligned}$$

利用  $y_0 = 0$ , 对上式从 0 到  $n-1$  求和, 得

$$\sum_{i=0}^{n-1} y_{i+1} = \sum_{i=0}^{n-1} y_i + \sum_{i=0}^{n-1} \frac{h}{2} [(ax_i + b) + (ax_{i+1} + b)]$$

所以

$$\begin{aligned} y_n &= \sum_{i=0}^{n-1} y_{i+1} - \sum_{i=0}^{n-1} y_i = \sum_{i=0}^{n-1} \frac{h}{2} [(ax_i + b) + (ax_{i+1} + b)] \\ &= \frac{ah}{2} \sum_{i=0}^{n-1} (x_i + x_{i+1}) + nbh \\ &= \frac{ah^2}{2} \sum_{i=0}^{n-1} [i + (i+1)] + nbh \\ &= \frac{ah^2}{2} \left[ \frac{1}{2} n(n-1) + \frac{1}{2} n(n+1) \right] + nbh \\ &= \frac{1}{2} a(nh)^2 + b(nh) = \frac{1}{2} ax_n^2 + bx_n = y(x_n) \end{aligned}$$

因而, 改进的欧拉方法能得到该初值问题的精确解。

例 9.3.2 初值问题  $y' = f(x, y), y(x_0) = y_0$ , 有如下方法

$$y_{n+1} = y_n + h(3f_n - f_{n-1})/2$$

式中  $f_n = f(x_n, y_n), f_{n-1} = f(x_{n-1}, y_{n-1})$ , 试说明此公式为几阶方法。

解 由泰勒展开式有

$$y(x_{n+1}) = y(x_n) + y'(x_n)h + \frac{y''(x_n)}{2!}h^2 + \frac{y'''(\xi_n)}{3!}h^3 \quad (9.3.4)$$

再把  $f(x_{n-1}, y(x_{n-1})) = y'(x_{n-1})$  在  $x_n$  处展开有

$$y'(x_{n-1}) = y'(x_n) + y''(x_n)(-h) + \frac{y'''(\eta_n)}{2!}(-h)^2$$

于是

$$\begin{aligned} y(x_n) + \frac{h}{2} [3y'(x_n) - y'(x_{n-1})] &= y(x_n) + \frac{h}{2} \left[ 3y'(x_n) - (y'(x_n) - hy''(x_n) + \frac{h^2}{2} y'''(\eta_n)) \right] \\ &= y(x_n) + hy'(x_n) + \frac{h^2}{2} y''(x_n) - \frac{h^3}{4} y'''(\eta_n) \end{aligned} \quad (9.3.5)$$

利用式 (9.3.4), 则所给公式的局部截断误差为

$$T_{n+1} = y(x_{n+1}) - \left\{ y(x_n) + \frac{h}{2} [3y'(x_n) - y'(x_{n-1})] \right\} = \frac{h^3}{12} y'''(\xi_n) = O(h^3)$$

根据定义, 可得计算公式



$$y_{n+1} = y_n + \frac{h}{2} [3f(x_n, y_n) - f(x_{n-1}, y_{n-1})]$$

是二阶方法。

## 9.4 龙格-库塔方法

### 9.4.1 龙格-库塔方法的基本思想

龙格-库塔 (Runge-Kutta) 方法是以德国数学家 C.Runge 及 M.W.kutta 的名字来命名的。它是求解常微分方程初值问题式 (9.1.1) 和式 (9.1.2) 的一类高精度的单步法。由定义 9.3.2 可知, 方法的精度与余项  $O(h^{p+1})$  有关。用一阶泰勒展开式推导出欧拉公式, 其余项为  $O(h^2)$ , 故是一阶方法。类似地, 若用  $p$  阶泰勒展开式

$$y_{n+1} = y(x_n) + hy'(x_n) + \frac{h^2}{2!} y''(x_n) + \cdots + \frac{h^p}{p!} y^{(p)}(x_n) + O(h^{p+1})$$

式中

$$y'(x) = f(x, y), y''(x) = f'_x(x, y) + f'_y(x, y)f(x, y) + \cdots$$

进行离散化, 所得数值公式必为  $p$  阶方法。由此, 我们能够想到, 通过提高泰勒展开式的阶数, 可以得到高精度的数值计算方法。从理论上讲, 只要微分方程 (9.1.1) 的解  $y(x)$  充分光滑, 泰勒展开方法就可以构造任意有限阶的计算公式。但事实上, 具体构造这种公式往往是相当困难的。因为求复合函数  $f(x, y(x))$  的高阶导数十分烦琐, 所以泰勒展开方法一般不直接使用。但是可以间接使用泰勒展开的方法, 求得高精度的数值计算方法。

首先, 我们对欧拉公式和改进的欧拉公式的形式进行进一步的分析。

如果将欧拉公式和改进的欧拉公式改成如下的形式

$$\text{欧拉公式} \quad \begin{cases} y_{n+1} = y_n + hk_1 \\ k_1 = f(x_n, y_n) \end{cases} \quad (9.4.1)$$

$$\text{改进的欧拉公式} \quad \begin{cases} y_{n+1} = y_n + h(\frac{1}{2}k_1 + \frac{1}{2}k_2) \\ k_1 = f(x_n, y_n) \\ k_2 = f(x_n + h, y_n + hk_1) \end{cases} \quad (9.4.2)$$

这两组公式都是用函数  $f(x, y)$  在某些点上的值的线性组合来计算  $y(x_{n+1})$  的近似值  $y_{n+1}$ 。欧拉公式每前进一步, 计算一次  $f(x, y)$  的值, 而且它是  $y(x_{n+1})$  在  $x_n$  处的一阶泰勒展开式, 因而是一阶方法。改进的欧拉公式每前进一步需要计算 2 次  $f(x, y)$  的值, 而且它在  $(x_n, y_n)$  处的泰勒展开式与  $y(x_{n+1})$  在  $x_n$  处的泰勒展开式的前三项完全相同, 因而是二阶方法。这就启发我们考虑用函数  $f(x, y)$  在若干点上的函数值的线性组合来构造数值公式。构造时要求计算公式在  $(x_n, y_n)$  处的泰勒展开式与微分方程的解  $y(x)$  在  $x_n$  处的泰勒展开式的前面若干项相同。从而使数值公式达到较高的精度。这样, 既避免了计算函数  $f(x, y)$  偏导数的困难, 又提高了数值计算方法的精度, 这就是龙格-库塔方法的基本思想。

### 9.4.2 二阶龙格-库塔方法的推导

龙格-库塔方法的一般形式为

$$\begin{cases} y_{n+1} = y_n + h \sum_{i=1}^p c_i k_i \\ k_1 = f(x_n, y_n) \\ \dots \\ k_i = f(x_n + a_i h, y_n + h \sum_{j=1}^{i-1} b_{ij} k_j) \end{cases} \quad (i = 2, 3, \dots, p) \quad (9.4.3)$$

式中,  $a_i, b_{ij}, c_i$  都是待定参数。确定它们的原则和方法是使计算公式在  $(x_n, y_n)$  处的泰勒展开式与微分方程的解  $y(x)$  在  $x_n$  处的泰勒展开式前面的项尽可能相同。从而使计算公式的精度尽可能地高。

常用的低阶二元函数的泰勒展开公式为

$$\begin{aligned} y'(x) &= f(x, y(x)) \\ y''(x) &= \left( \frac{\partial f}{\partial x} + f \frac{\partial f}{\partial y} \right) (x, y(x)) \\ y'''(x) &= \left( \frac{\partial^2 f}{\partial x^2} + 2f \frac{\partial^2 f}{\partial x \partial y} + f^2 \frac{\partial^2 f}{\partial y^2} \right) (x, y(x)) + \left( \frac{\partial f}{\partial x} + f \frac{\partial f}{\partial y} \right) (x, y(x)) \frac{\partial f}{\partial y} (x, y(x)) \\ &= \left( \frac{\partial^2 f}{\partial x^2} + 2f \frac{\partial^2 f}{\partial x \partial y} + f^2 \frac{\partial^2 f}{\partial y^2} \right) (x, y(x)) + y''(x) \frac{\partial f}{\partial y} (x, y(x)) \end{aligned}$$

下面推导二阶龙格-库塔方法。

设  $p=2$ , 此时, 计算公式为

$$\begin{cases} y_{n+1} = y_n + h(c_1 k_1 + c_2 k_2) \\ k_1 = f(x_n, y_n) \\ k_2 = f(x_n + a_2 h, y_n + h b_{21} k_1) \end{cases} \quad (9.4.4)$$

将式 (9.4.4) 在  $(x_n, y_n)$  处泰勒展开

$$\begin{aligned} y_{n+1} &= y_n + h[c_1 f(x_n, y_n) + c_2 f(x_n + a_2 h, y_n + h b_{21} f(x_n, y_n))] \\ &= y_n + h[c_1 f(x_n, y_n) + c_2 (f(x_n, y_n) + a_2 h f'_x(x_n, y_n) + b_{21} h f'_y(x_n, y_n) f(x_n, y_n))] + O(h^3) \quad (9.4.5) \\ &= y_n + (c_1 + c_2) f(x_n, y_n) h + c_2 [a_2 f'_x(x_n, y_n) + b_{21} f'_y(x_n, y_n) f(x_n, y_n)] h^2 + O(h^3) \end{aligned}$$

$y(x_{n+1})$  在  $x_n$  处的泰勒展开式为

$$\begin{aligned} y(x_{n+1}) &= y(x_n) + h y'(x_n) + \frac{h^2}{2} y''(x_n) + O(h^3) \\ &= y_n + f(x_n, y_n) h + \frac{h^2}{2} [f'_x(x_n, y_n) + f'_y(x_n, y_n) f(x_n, y_n)] + O(h^3) \end{aligned} \quad (9.4.6)$$

要使式 (9.4.4) 的局部截断误差为  $O(h^3)$ , 则应要求式 (9.4.5) 和式 (9.4.6) 的前三项相同,

$$\text{因此有} \quad \begin{cases} c_1 + c_2 = 1 \\ c_2 a_2 = \frac{1}{2} \\ c_2 b_{21} = \frac{1}{2} \end{cases} \quad (9.4.7)$$

式 (9.4.7) 有 4 个未知数, 3 个方程, 其中一个是自由参数, 式 (9.4.7) 有无穷多个解, 从而得到一组二阶龙格-库塔方法, 它们的局部截断误差均为  $O(h^3)$ 。

例如, 取  $c_1 = c_2 = \frac{1}{2}$ ,  $a_2 = b_{21} = 1$ , 计算公式为

$$\begin{cases} y_{n+1} = y_n + \frac{h}{2}(k_1 + k_2) \\ k_1 = f(x_n, y_n) \\ k_2 = f(x_n + h, y_n + hk_1) \end{cases}$$

它与式 (9.4.2) 相同, 这就是改进的欧拉公式。

取  $c_1 = 0, c_2 = 1, a_2 = b_{21} = \frac{1}{2}$ , 计算公式为

$$\begin{cases} y_{n+1} = y_n + hk_2 \\ k_1 = f(x_n, y_n) \\ k_2 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1) \end{cases} \quad (9.4.8)$$

式 (9.4.8) 称为中点公式。

取  $c_1 = \frac{1}{4}, c_2 = \frac{3}{4}, a_2 = \frac{2}{3}, b_2 = \frac{2}{3}$ , 计算公式为

$$\begin{cases} y_{n+1} = y_n + h(\frac{1}{4}k_1 + \frac{3}{4}k_2) \\ k_1 = f(x_n, y_n) \\ k_2 = f(x_n + \frac{2}{3}h, y_n + \frac{2}{3}hk_1) \end{cases} \quad (9.4.9)$$

式 (9.4.9) 称为 Heun 公式。

**例 9.4.1** 证明 Heun 公式  $y_{n+1} = y_n + \frac{h}{4} \left[ f(x_n, y_n) + 3f(x_n + \frac{2}{3}h, y_n + \frac{2}{3}hf(x_n, y_n)) \right]$  是二阶的, 并给出它的局部截断误差主项。

**证明** 公式的局部截断误差为  $T_{n+1} = y(x_{n+1}) - y(x_n) - \frac{h}{4} \left[ y'(x_n) + 3f(x_n + \frac{2}{3}h, y_n + \frac{2}{3}hf_n) \right]$

将右端泰勒展开, 注意

$$\begin{aligned} y(x_{n+1}) &= y(x_n) + hy'(x_n) + \frac{h^2}{2}y''(x_n) + \frac{h^3}{3!}y'''(x_n) + O(h^4) \\ f(x_n + \frac{2}{3}h, y_n + \frac{2}{3}hf_n) &= y'(x_n) + \frac{2}{3}hy''(x_n) + \frac{1}{2}(\frac{2}{3}h)^2(f''_{xx} + 2ff''_{xy} + f^2f''_{yy})_{x_n} + O(h^4) \end{aligned}$$

于是

$$T_{n+1} = \frac{h^3}{6} \left[ y'''(x_n) - (f''_{xx} + 2ff''_{xy} + f^2f''_{yy})_{x_n} \right] + O(h^4) = \frac{h^3}{6} y''(x_n) f'_y(x_n, y_n) + O(h^4)$$

故方法是二阶的, 且局部截断误差主项为  $\frac{h^3}{6} y''(x_n) f'_y$ 。

**例 9.4.2** 证明下列公式是三阶龙格-库塔公式。

$$\begin{cases} y_{n+1} = y_n + \frac{h}{9}(2k_1 + 3k_2 + 4k_3) \\ k_1 = f(x_n, y_n) \\ k_2 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1) \\ k_3 = f(x_n + \frac{3}{4}h, y_n + \frac{3}{4}hk_2) \end{cases}$$

**证明** 所给公式是用区间  $[x_n, x_{n+1}]$  内三点  $x_n, x_n + \frac{h}{2}, x_n + \frac{3}{4}h$  的斜率值  $k_1, k_2, k_3$  加权平均生成该区间内的平均斜率, 考察相应的离散关系式

$$\begin{cases} y(x_{n+1}) \approx y(x_n) + \frac{h}{9}(2k_1 + 3k_2 + 4k_3) \\ k_1 = f(x_n, y(x_n)) \\ k_2 = f(x_n + \frac{h}{2}, y(x_n) + \frac{h}{2}k_1) \\ k_3 = f(x_n + \frac{3}{4}h, y(x_n) + \frac{3}{4}hk_2) \end{cases}$$

将  $k_1, k_2, k_3$  泰勒展开, 采用缩记符号, 得

$$\begin{aligned} k_1 &= y'_n \\ k_2 &= f_n + \frac{1}{2}h \left( \frac{\partial f}{\partial x} + k_1 \frac{\partial f}{\partial y} \right)_n + \frac{1}{2} \left( \frac{1}{2}h \right)^2 \left( \frac{\partial^2 f}{\partial x^2} + 2k_1 \frac{\partial^2 f}{\partial x \partial y} + k_1^2 \frac{\partial^2 f}{\partial y^2} \right)_n + O(h^3) \end{aligned}$$

从而利用求导公式可得

$$k_2 = y'_n + \frac{1}{2}hy''_n + \frac{h^2}{8}[y'''_n - y''_n \left( \frac{\partial f}{\partial y} \right)_n] + O(h^3)$$

此外, 类似地, 得

$$\begin{aligned} k_3 &= f_n + \frac{3}{4}h \left( \frac{\partial f}{\partial x} + k_2 \frac{\partial f}{\partial y} \right)_n + \frac{1}{2} \left( \frac{3}{4}h \right)^2 \left( \frac{\partial^2 f}{\partial x^2} + 2k_2 \frac{\partial^2 f}{\partial x \partial y} + k_2^2 \frac{\partial^2 f}{\partial y^2} \right)_n + O(h^3) \\ &= y'_n + \frac{3}{4}hy''_n + \frac{1}{2} \left( \frac{3}{4}h \right)^2 y'''_n - \frac{9h^2}{32}y''_n \left( \frac{\partial f}{\partial y} \right)_n + O(h^3) \end{aligned}$$

代入离散关系式右端, 并记所得结果为  $y_{n+1}^*$ , 则有

$$y_{n+1}^* = y_n + hy'_n + \frac{h^2}{2}y''_n + \frac{h^3}{6}y'''_n + O(h^4)$$

它同  $y(x_{n+1})$  的泰勒展开式符合到  $h^3$  项, 故所给公式都是三阶方法。

### 9.4.3 四阶经典龙格-库塔方法

9.4.2 节我们推导了二阶和三阶龙格-库塔方法, 类似地, 可以用同样的方法推导更高阶的龙格-库塔方法。在实际当中, 最常用的是四阶经典龙格-库塔方法, 其形式为

$$\begin{cases} y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 = f(x_n, y_n) \\ k_2 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1) \\ k_3 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2) \\ k_4 = f(x_n + h, y_n + hk_3) \end{cases} \quad (9.4.10)$$

四阶经典龙格-库塔方法的算法框图如图 9.4.1 所示。

输入 $x_0, y_0, N, h$
对于 $i = 1, 2, 3, \dots, N$ 计算
$x_1 = x_0 + h, \quad k_1 = f(x_0, y_0)$ $k_2 = f(x_0 + \frac{h}{2}, y_0 + \frac{h}{2} k_1)$ $k_3 = f(x_0 + \frac{h}{2}, y_0 + \frac{h}{2} k_2), \quad k_4 = f(x_1, y_0 + h k_3)$ $y_1 = y_0 + \frac{h}{6} (k_1 + 2k_2 + 2k_3 + k_4)$
输出 $(x_1, y_1)$ , $x_0 = x_1, y_0 = y_1$

图 9.4.1 四阶经典龙格-库塔方法的算法框图

例 9.4.3 用四阶经典龙格-库塔方法，求解例 9.2.2，取步长  $h = 0.2$ 。

解 例 9.2.2 的四阶经典龙格-库塔方法的计算公式为

$$\begin{cases} y_{n+1} = y_n + \frac{0.2}{6} (k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 = 2x_n y_n \\ k_2 = 2(x_n + \frac{h}{2})(y_n + \frac{h}{2} k_1) \\ k_3 = 2(x_n + \frac{h}{2})(y_n + \frac{h}{2} k_2) \\ k_4 = 2(x_n + h)(y_n + h k_3) \end{cases}$$

计算结果见表 9.4.1。

表 9.4.1 例 9.4.3 的计算结果

$x_n$	$y_n$	$k_1$	$k_2$	$k_3$	$k_4$
0	1	0	0.2	0.204	0.416 32
0.2	1.040 810 7	0.416 324 3	0.649 465 9	0.663 454 4	0.938 801 3
0.4	1.01 735 096	0.938 807 7	1.267 390 4	1.300 248 6	1.720 271 2
0.6	1.04 333 215	1.719 985 8	2.474 481	2.321 292 8	3.361 281
0.8	1.896 441 4	3.034 306 2	3.959 769 6	4.163 531	5.443 424 0
1	2.718 107 3				

例 9.4.4 用欧拉方法、改进的欧拉方法和四阶经典龙格-库塔方法，求解初值问题

$$\begin{cases} y' = x - y & 0 \leq x \leq 1 \\ y(0) = 0 \end{cases} \quad (9.4.11)$$

的数值解。

解 此问题的精确解为  $y = x + e^{-x} - 1$

对初值问题式 (9.4.11) 的欧拉公式 ( $h = 0.1, N = 10$ ) 为

$$\begin{aligned} y_{n+1} &= y_n + h(x_n - y_n) \\ &= (1 - 0.1)y_n + x_n \\ &= 0.9y_n + x_n \end{aligned}$$

对初值问题式 (9.4.11) 的改进的欧拉公式 ( $h = 0.1, N = 10$ ) 为

$$\begin{aligned}
y_{n+1} &= y_n + \frac{h}{2}[x_n - y_n + x_{n+1} - y_n + h(x_n - y_n)] \\
&= (1 - h + \frac{h^2}{2})y_n + \frac{h}{2}(1 - h)x_n + \frac{h}{2}x_{n+1} \\
&= 0.9005y_n + 0.045x_n + 0.05x_{n+1}
\end{aligned}$$

对初值问题式 (9.4.11) 的四阶经典龙格-库塔方法的公式 ( $h = 0.2, N = 5$ ) 为

$$\begin{cases}
y_{n+1} = y_n + \frac{0.2}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\
k_1 = x_n - y_n \\
k_2 = x_n + \frac{h}{2} - (y_n + \frac{h}{2}k_1) = 0.9(x_n - y_n) + 0.1 \\
k_3 = x_n + \frac{h}{2} - (y_n + \frac{h}{2}k_2) = 0.91(x_n - y_n) + 0.09 \\
k_4 = x_n + h - (y_n + hk_3) = 0.818(x_n - y_n) + 0.182
\end{cases}$$

代入初值  $x_0 = 0, y_0 = 0$ ，计算结果见表 9.4.2。

表 9.4.2 例 9.4.4 的计算结果

$x_n$	欧拉公式 $y_n$	改进的欧拉公式 $y_n$	四阶龙格-库塔公式 $y_n$	精确解 $y(x_n)$
0	0.000 000	0.000 000	0.000 000	0.000 000
0.1	0.000 000	0.005 000		0.004 837
0.2	0.010 000	0.019 025	0.018 733	0.001 873 1
0.3	0.029 000	0.041 218		0.041 818
0.4	0.056 100	0.070 802	0.070 324	0.070 320
0.5	0.090 490	0.107 076		0.010 653 1
0.6	0.131 441	0.149 404	0.114 881 7	0.148 812
0.7	0.178 297	0.197 211		0.196 585
0.8	0.230 467	0.249 976	0.249 335	0.249 329
0.9	0.287 420	0.307 228		0.306 570
1.0	0.348 678	0.368 541	0.367 886	0.367 876

从例 9.2.2、例 9.4.3 和例 9.4.4 的计算结果来看，显然四阶经典龙格-库塔方法的精度要高得多。与改进的欧拉公式相比，四阶方法虽然每步需要计算 4 个函数值，但由于步长放大了一倍，计算量与改进的欧拉公式几乎相同，但精度明显提高。需要指出的是，四阶及四阶以下的龙格-库塔方法的阶数与每前进一步计算函数  $f(x, y)$  的次数  $p$  是一致的。但更高阶的情形则不然，例如  $p = 5$  时，龙格-库塔公式的最高阶数仍为 4； $p = 6$  时，龙格-库塔公式的最高阶数为 5。由于计算量较大，在实际应用中，很少使用更高次的龙格-库塔方法。四阶龙格-库塔方法已可满足对精度的要求。另外，由于龙格-库塔方法的推导是基于泰勒展开的，因而它要求所求微分方程初值问题的解具有较好的光滑性质。如果解的光滑性差，则用四阶龙格-库塔方法求初值问题的数值解的效果可能不如改进的欧拉公式，因此，在实际计算时，需要根据问题的具体情况来选择合适的算法。

## 9.5 单步法的收敛性和稳定性

收敛性与稳定性从两个不同的角度描述了微分方程数值解法的实用价值。收敛性反映计算

公式本身的截断误差对计算结果的影响；稳定性反映某一公式在计算过程中出现的误差对计算结果的影响。只有既收敛又稳定的方法，才可能提供比较可靠的计算结果。

### 9.5.1 单步法的收敛性

微分方程数值解法的基本思想是：通过某种离散化手段，将微分方程转化为差分方程（代数方程）来求解。这种转化是否合理，还要看差分问题的解  $y_n$ ，当  $h \rightarrow 0$  时是否会收敛到微分方程的准确解。需要注意的是，如果只考虑  $h \rightarrow 0$ ，那么节点  $x_n = x_0 + nh$  对固定的  $n$  将趋向于  $x_0$ ，这时讨论收敛性是没有意义的，因此当  $h \rightarrow 0$  时，同时要求  $n \rightarrow \infty$  才合理。

**定义 9.5.1** 若求微分方程的一种数值计算方法对于任意固定的  $x_n = x_0 + nh$ ，当  $h \rightarrow 0$ （同时  $n \rightarrow \infty$ ）时，有  $y_n \rightarrow y(x_n)$ ，则称该方法是收敛的。

为了讨论方便，将微分方程（9.1.1）中的  $f(x, y)$  在解域内某一点  $(a, b)$  进行泰勒展开，并局部线性化

$$\begin{aligned} y' = f(x, y) &= f(a, b) + (x - a)f_x(a, b) + (y - b)f_y(a, b) + \cdots \\ &= f_y(a, b)y + c_1x + c_2 + \cdots \end{aligned}$$

忽略高阶项，令

$$\lambda = f_y(a, b) \text{ 和 } y = u - \frac{c_1}{\lambda}x - \frac{c_2}{\lambda^2} - \frac{c_2}{\lambda}$$

对上式进行变量置换，得  $u' = \lambda u$ ，因此对一般形式的一阶微分方程，总能化简成如下的模型方程

$$y' = \lambda y$$

为保证微分方程初值问题的求解，必须要求  $\lambda = f_y < 0$ 。

本节讨论的单步法的收敛性和稳定性都是通过模型方程来讨论的。

讨论下面模型方程的初值问题

$$\begin{cases} y' = \lambda y & (\lambda < 0) \\ y(0) = y_0 \end{cases} \quad (9.5.1)$$

$$(9.5.2)$$

首先，考察欧拉方法的收敛性，将欧拉方法应用于方程  $y' = \lambda y$ ，得

$$y_{n+1} = y_n + hf(x_n, y_n) = y_n + h\lambda y_n = (1 + h\lambda)y_n$$

所以数值解

$$y_n = (1 + h\lambda)y_{n-1} = (1 + h\lambda)^2 y_{n-2} = \cdots = y_0(1 + h\lambda)^n$$

由于  $x_0 = 0, x_n = nh$ ，有

$$y_n = y_0(1 + \lambda h)^{\frac{x_n}{h}} = y_0[(1 + \lambda h)^{\frac{1}{\lambda h}}]^{\lambda x_n}$$

注意到，当  $h \rightarrow 0$  时

$$(1 + \lambda h)^{\frac{1}{\lambda h}} \rightarrow e$$

所以，当  $h \rightarrow 0$  时，有

$$y_n \rightarrow y_0 e^{\lambda x_n}$$

而另一方面，解微分方程（9.5.1）得

$$\frac{dy}{dx} = \lambda y, \quad \frac{1}{y} dy = \lambda dx$$

两边积分  $\int \frac{1}{y} dy = \int \lambda dx$ , 得

$$\ln y = \lambda x + c$$

即

$$y(x) = e^{\lambda x + c} = e^c e^{\lambda x}$$

由初始条件  $y(0) = y_0$ , 得

$$e^c = y_0$$

于是, 微分方程初值问题式 (9.5.1) 和式 (9.5.2) 的精确解为

$$y(x) = y_0 e^{\lambda x}$$

所以

$$y(x_n) = y_0 e^{\lambda x_n}$$

综合上述结论可得, 当  $h \rightarrow 0$  (同时  $n \rightarrow \infty$ ) 时

$$y_n \rightarrow y(x_n)$$

这说明欧拉方法是收敛的。

**例 9.5.1** 用梯形方法解初值问题  $\begin{cases} y' + y = 0 \\ y(0) = 1 \end{cases}$ , 证明其近似解为  $y_n = \left(\frac{2-h}{2+h}\right)^n$ , 并证明  $h \rightarrow 0$

(同时  $n \rightarrow \infty$ ) 时, 它收敛于原初值问题的准确解  $y = e^{-x}$ 。

**证明** 已知梯形公式为

$$y_{n+1} = y_n + \frac{h}{2} [f(x_n, y_n) + f(x_{n+1}, y_{n+1})]$$

将  $f(x, y) = -y$  代入该计算公式, 得  $y_{n+1} = y_n + \frac{h}{2} (-y_n - y_{n+1})$ , 则有

$$y_n = y_{n-1} + \frac{h}{2} (-y_{n-1} - y_n)$$

经整理得  $y_n = \left(\frac{2-h}{2+h}\right) y_{n-1} = \left(\frac{2-h}{2+h}\right)^2 y_{n-2} = \cdots = \left(\frac{2-h}{2+h}\right)^n y_0$

因为  $y_0 = 1$ , 所以证明了该问题用梯形公式求得近似解为  $y_n = \left(\frac{2-h}{2+h}\right)^n$

当  $h \rightarrow 0$  时对上式取极限, 并注意到  $x_n = nh$ , 有

$$\lim_{h \rightarrow 0} y_n = \lim_{h \rightarrow 0} \left(\frac{2-h}{2+h}\right)^n = \lim_{h \rightarrow 0} \left(1 - \frac{2h}{2+h}\right)^{\frac{x_n}{h}} = \lim_{h \rightarrow 0} \left(1 - \frac{2h}{2+h}\right)^{\left(-\frac{2+h}{2h}\right)\left(-\frac{2x_n}{2+h}\right)} = e^{-x_n}$$

这说明, 对于本题所研究的初值问题, 梯形公式是收敛的, 且收敛于问题的精确解。

下面进一步考察一般的单步法。

所谓单步法, 就是在计算  $y_{n+1}$  时只用到它前一步的信息  $y_n$ 。显式单步法的共同特征是, 它们都是将  $y_n$  加上某种形式的增量得出  $y_{n+1}$ , 其计算公式的形式为

$$y_{n+1} = y_n + h\varphi(x_n, y_n, h) \quad (9.5.3)$$

式中,  $\varphi(x_n, y_n, h)$  称为增量函数。

不同的单步法, 对应不同的增量函数, 例如, 欧拉公式的增量函数为  $\varphi = f(x, y)$ , 而改进的欧拉公式 (9.2.4) 的增量函数为

$$\varphi = \frac{1}{2} [f(x, y) + f(x+h, y+hf(x, y))] \quad (9.5.4)$$



关于单步法有下述收敛性定理。

**定理 9.5.1** 假设单步法式(9.5.3)具有  $p$  阶精度, 且增量函数  $\varphi(x, y, h)$  关于  $y$  满足 Lipschitz 条件

$$|\varphi(x, y, h) - \varphi(x, \bar{y}, h)| \leq L_\varphi |y - \bar{y}| \quad (9.5.5)$$

且设初值  $y_0$  是准确的, 即  $y_0 = y(x_0)$ , 则单步法式 (9.5.3) 的整体截断误差为

$$y(x_n) - y_n = O(h^p)$$

定理证明略。

根据这一定理, 判断单步法式 (9.5.3) 的收敛性, 关键是验证增量函数  $\varphi$  是否满足 Lipschitz 条件式 (9.5.5)。

**例 9.5.2** 已知初值问题  $y' = ax + b$ ,  $y(0) = 0$  的精确解是  $y(x) = \frac{a}{2}x^2 + bx$ , 证明用欧拉法以  $h$

为步长所得近似解  $y_n$  的整体误差为  $\varepsilon_n = y(x_n) - y_n = \frac{1}{2}ahx_n$ 。

**证明** 由欧拉公式解  $y' = ax + b$ , 得  $y_n = y_{n-1} + h(ax_{n-1} + b)$

由  $y(0) = 0 = y_0$ , 得

$$y_1 = bh$$

$$y_2 = y_1 + h(ax_1 + b) = 2bh + ahx_1$$

$$y_3 = y_2 + h(ax_2 + b) = 3bh + ah(x_1 + x_2)$$

...

$$y_n = y_{n-1} + h(ax_{n-1} + b) = nbh + ah(x_1 + x_2 + \cdots + x_{n-1})$$

因为  $x_n = nh$ , 于是

$$y_n = bx_n + ah^2 [1 + 2 + \cdots + (n-1)] = bx_n + ah^2 \frac{(n-1)n}{2} = \frac{1}{2}ax_{n-1}x_n + bx_n$$

所以整体误差为  $\varepsilon_n = y(x_n) - y_n = \frac{a}{2}(x_n^2 - x_{n-1}x_n) = \frac{ahx_n}{2}$

证毕。

对于改进的欧拉公式, 其增量函数为式 (9.5.4) 所以有

$$|\varphi(x, y, h) - \varphi(x, \bar{y}, h)| \leq \frac{1}{2} [|f(x, y) - f(x, \bar{y})| + |f(x+h, y+hf(x, y)) - f(x+h, \bar{y}+hf(x, \bar{y}))|]$$

假设右端函数  $f$  关于  $y$  满足 Lipschitz 条件, 记 Lipschitz 常数为  $L$ , 则由上式可得

$$|\varphi(x, y, h) - \varphi(x, \bar{y}, h)| \leq L(1 + \frac{h}{2}L) |y - \bar{y}|$$

不妨限定  $h < h_0$  ( $h_0$  为固定的常数), 上式表明  $\varphi$  关于  $y$  的 Lipschitz 常数为

$$L_\varphi = L(1 + \frac{h_0}{2}L)$$

由此判断, 改进的欧拉方法也是收敛的。用同样的方法可以验证, 龙格-库塔方法也是收敛的。

## 9.5.2 单步法的稳定性

本节讨论的稳定性, 不是常微分方程初值问题式 (9.1.1) 和式 (9.1.2) 本身的稳定性, 而是指数值计算方法的稳定性, 即数值稳定性。

一种计算方法, 即使是收敛的, 初始值一般都带有误差, 同时, 在计算过程中还常常产生

舍入误差，这些误差又必然会传播下去，对后续的计算结果都将产生影响，数值稳定性问题是讨论这种误差的积累和传播能否得到控制的问题。

**定义 9.5.2 (方法的稳定性)** 若用某种计算方法计算  $y_n$  时，所得到的实际计算结果为  $\tilde{y}_n$ ，且由扰动  $\delta_n = |y_n - \tilde{y}_n|$  引起以后各节点  $y_m$  ( $m > n$ ) 的扰动为  $\delta_m$ ，如果总有  $|\delta_m| \leq |\delta_n|$ ，则称该方法是稳定的。

一种计算方法是否稳定，不仅与该计算方法本身有关，而且还与微分方程的右端函数  $f(x, y)$  及步长  $h$  有关，因此稳定性问题比较复杂。为了简化讨论，只考虑模型方程

$$y' = \lambda y \quad (\lambda < 0)$$

首先，讨论欧拉方法的稳定性。将欧拉公式应用于模型方程  $y' = \lambda y$  为

$$y_{n+1} = (1 + h\lambda)y_n \quad (9.5.6)$$

设在节点  $y_n$  处有一个扰动值  $\delta_n$ ，它的传播使节点  $y_{n+1}$  处产生一个扰动值  $\delta_{n+1}$ ，假设用  $\tilde{y}_n = y_n + \delta_n$  按欧拉公式得出  $\tilde{y}_{n+1} = y_{n+1} + \delta_{n+1}$  的计算过程中不再产生新的误差，则扰动值满足

$$\delta_{n+1} = (1 + h\lambda)\delta_n$$

可见，扰动值满足原来的差分方程 (9.5.6)，因此，如果原差分方程的解不增长，即有

$$|y_{n+1}| \leq |y_n|$$

就能保证欧拉方法的稳定性。令

$$E(h\lambda) = 1 + h\lambda$$

显然，为了保证差分方程 (9.5.6) 的解不增长，必须选取  $h$  充分小，使

$$|E(h\lambda)| = |1 + h\lambda| \leq 1$$

这表明欧拉方法是条件稳定的，稳定性的区间为

$$-2 \leq h\lambda < 0$$

下面再讨论梯形公式的稳定性。模型方程  $y' = \lambda y$  的梯形公式为

$$y_{n+1} = y_n + \frac{h}{2}(\lambda y_n + \lambda y_{n+1})$$

整理后得

$$y_{n+1} = \frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}} y_n$$

令

$$E(h\lambda) = \frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}}$$

同理，扰动值  $\delta_n$  满足  $\delta_{n+1} = E(h\lambda)\delta_n$ ，因为  $\lambda < 0$ ，所以对任何  $\lambda h$  都有  $|E(h\lambda)| < 1$ ，这说明梯形公式是无条件稳定的。

事实上，任何一种单步法，应用于模型方程 (9.5.1) (其中  $\lambda = f_y < 0$ )，均有

$$y_{n+1} = E(h\lambda)y_n \quad (9.5.7)$$

对于不同的单步法， $E(h\lambda)$  有不同的表达式。例如，对于模型方程  $y' = \lambda y$ ，改进的欧拉公式为

$$\begin{aligned}
 y_{n+1} &= y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y_n + hf(x_n, y_n))] \\
 &= y_n + \frac{h}{2}[\lambda y_n + \lambda(y_n + h\lambda y_n)] \\
 &= y_n + \frac{h}{2}[\lambda y_n + \lambda y_n + h\lambda^2 y_n] \\
 &= (1 + h\lambda + \frac{h^2\lambda^2}{2})y_n
 \end{aligned}$$

所以

$$E(h\lambda) = 1 + h\lambda + \frac{(\lambda h)^2}{2}$$

由  $|E(h\lambda)| \leq 1$  可得  $-2 \leq h\lambda < 0$ 。

**定义 9.5.3 (方法的稳定区域)** 若式 (9.5.7) 中的  $|E(h\lambda)| \leq 1$ ，则说该单步法是稳定的。在复平面上， $h\lambda$  满足  $|E(h\lambda)| \leq 1$  的区域，称为方法的稳定区域，它与实轴的交点称为稳定区间。

**例 9.5.3** 对模型方程  $y' = \lambda y (\lambda < 0)$ ，证明隐式欧拉公式对任何步长  $h > 0$  都绝对稳定。

**证明** 将隐式欧拉公式  $y_{n+1} = y_n + hf(x_{n+1}, y_{n+1})$  应用于方程  $y' = \lambda y$ ，得

$$y_{n+1} = y_n + h\lambda y_{n+1}$$

即

$$y_{n+1} = \frac{1}{1 - h\lambda} y_n, \lambda < 0$$

由  $1 - h\lambda > 1$  知  $\left| \frac{1}{1 - h\lambda} \right| < 1$ ，故对任何  $h > 0$ ，方法绝对稳定。

表 9.5.1 列出了常用单步法的  $E(h\lambda)$  表达式和稳定区间。

表 9.5.1 单步法的表达式和稳定区间

方法	$E(h\lambda)$	稳定区间
欧拉法	$1 + h\lambda$	$-2 \leq h\lambda < 0$
改进的欧拉法	$1 + h\lambda + \frac{(\lambda h)^2}{2}$	$-2 \leq h\lambda < 0$
四阶龙格-库塔法	$1 + h\lambda + \frac{(h\lambda)^2}{2!} + \frac{(h\lambda)^3}{3!} + \frac{(h\lambda)^4}{4!}$	$-2.785 \leq \lambda h \leq 0$
隐式欧拉法	$\frac{1}{1 - \lambda h}$	$-\infty \leq \lambda h \leq 0$
梯形公式	$\frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}}$	$-\infty \leq \lambda h \leq 0$

**例 9.5.4** 对于初值问题  $y' = -100(y - x^2) + 2x, y(0) = 1$ ：

- (1) 用欧拉法求解，步长  $h$  应取在什么范围内迭代计算才稳定？
- (2) 若用梯形法求解，对步长  $h$  有无限制？
- (3) 若用四阶龙格-库塔方法求解，步长  $h$  如何选取？

**解** 因为  $f'_y = -100$ ，故由稳定区间要求可知：

- (1) 用欧拉法求解时， $0 < h \leq \frac{2}{100} = 0.02$ 。

(2) 用梯形法求解时, 稳定区间为  $0 < h < +\infty$ 。又因为  $f$  对  $y$  是线性的, 所以对  $h$  无限制。

(3) 用四阶龙格-库塔方法时,  $0 < h \leq \frac{2.785}{100} = 0.02785$ 。

例 9.5.5 对初值问题

$$\begin{cases} \frac{dy}{dx} = -20y & x \in [0, 1] \\ y(0) = 1 \end{cases}$$

分别取  $h = 0.1$ ,  $h = 0.2$ , 用经典四阶龙格-库塔方法求数值解。

解 直接用经典四阶龙格-库塔方法的公式 (9.4.10) 计算, 因为精确解为  $y(x) = e^{-20x}$ , 所以其计算误差  $|y_n - y(x_n)|$  见表 9.5.2。

表 9.5.2 各步数值解的计算误差

$x_n$	0.2	0.4	0.6	0.8	1.0
$h = 0.1$	0.092 795	0.012 010	0.001 366	0.000 152	0.000 017
$h = 0.2$	4.98	25.0	125.0	625.0	3125.0

从表 9.5.2 可以看出, 当步长  $h = 0.1$  时, 各步数值解的误差较小且逐渐衰减; 当步长  $h = 0.2$  时, 各步数值解的误差较大且迅速增长, 以致失去控制。产生这种现象的原因是, 当  $h = 0.1$  时,  $\lambda h = -20 \times 0.1 = -2$ , 落在稳定区间  $[-2.78, 0]$  中; 而当  $h = 0.2$  时,  $\lambda h = -20 \times 0.2 = -4$ , 却不在稳定区间内。因此, 选择步长时, 不仅要考虑截断误差, 还应考虑计算方法的稳定性。

## 本章小结

本章讨论了求解常微分方程初值问题的一些数值解法。欧拉方法和龙格-库塔方法都是将微分方程离散化为差分方程求解, 是步进式的方法。欧拉方法由于精度低, 在实际中很少使用, 但由于公式简单、直观, 对学习其他方法具有启示作用。四阶龙格-库塔公式由于其精度高、易于编程, 因此在实际中得到了广泛的应用, 但其缺点是要求右端函数  $f(x, y)$  具有很好的光滑性, 且每步都需要计算 4 次函数值, 计算量较大。

数值计算方法的收敛性和稳定性是判断方法好坏的两个重要指标, 一般只要增量函数  $\varphi(x, y, h)$  关于  $y$  满足 Lipschitz 条件, 单步法都是收敛的, 隐式单步法是绝对稳定的, 而显式单步法有自己的稳定区域, 在选取步长时, 应该格外注意。

## 习题 9

9.1 用欧拉法、隐式欧拉法、梯形法解初值问题  $y' = -y + x + 1$ ,  $y(0) = 1$ , 取  $h = 0.1$ , 计算到  $x = 0.5$ , 并与精确解  $y(x) = e^{-x} + x$  比较。

9.2 用改进欧拉法求初值问题  $y' = -y + x + 1$ ,  $y(0) = 1$ , 取  $h = 0.1$ , 计算到  $x = 0.5$ , 并与欧拉法和梯形法比较误差的大小。

9.3 用经典四阶龙格-库塔方法解初值问题  $y' = -y + x + 1$ ,  $y(0) = 1$ , 仍取  $h = 0.1$ , 计算到  $x = 0.5$ , 并与改进的欧拉法和梯形法在  $x_5 = 0.5$  处比较其误差大小。

#### 9.4 用欧拉方法求解初值问题

$$\begin{cases} y' = y - \frac{2x}{y} \\ y(0) = 1 \end{cases} \quad (0 < x < 1)$$

#### 9.5 用改进的欧拉方法求解初值问题

$$\begin{cases} y' = y - \frac{2x}{y} \\ y(0) = 1 \end{cases} \quad (0 < x < 1)$$

#### 9.6 用四阶龙格-库塔经典公式

$$\begin{cases} y_{n+1} = y_n + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \\ K_1 = f(x_n, y_n) \\ K_2 = f(x_{n+\frac{1}{2}}, y_n + \frac{h}{2}K_1) \\ K_3 = f(x_{n+\frac{1}{2}}, y_n + \frac{h}{2}K_2) \\ K_4 = f(x_{n+1}, y_n + hK_3) \end{cases}$$

求解初值问题

$$\begin{cases} y' = y - \frac{2x}{y} \\ y(0) = 1 \end{cases} \quad (0 < x < 1)$$

取步长  $h=0.2$ ，从  $x=0$  到  $x=1$ 。

9.7 对于初值问题  $\begin{cases} y' = -1000(y - g(x)) + g'(x) \\ y(0) = y_0 \end{cases}$ ，式中  $g(x)$  为已知函数，其精确解为：

$y(x) = g(x)$ 。

(1) 若用显式欧拉法求解，从稳定性考虑步长应在什么范围内选取？

(2) 若用隐式欧拉法求解，从稳定性考虑步长有没有限制，为什么？

(3) 若  $g(x)$  为不超过一次的多项式，用显式欧拉法求解此问题时，从精确解考虑，步长的选择有无限制？为什么？

9.8 对初值问题  $y' = f(x, y)$ ， $y(a) = y_0$ ， $x_0 = a$ ， $a \leq x \leq b$ ， $x_n = x_0 + nh$ ，试用数值积分在区间  $[x_n, x_{n+1}]$  或  $[x_{n-1}, x_{n+1}]$  内对  $y' = f(x, y)$  两边积分，分别导出以下公式

(1) 梯形公式 
$$y_{n+1} = y_n + \frac{h}{2}(f_n + f_{n+1})$$

(2) 中点公式 
$$y_{n+1} = y_{n-1} + 2hf_n$$

(3) 辛普生公式 
$$y_{n+1} = y_{n-1} + \frac{h}{3}(4f_n + f_{n+1} + f_{n-1})$$

并给出各公式的局部截断误差。

9.9 对模型方程  $y' = \lambda y$  ( $\lambda < 0$ ) 求改进欧拉公式的稳定区间。

9.10 证明：解初值问题  $y' = f(x, y)$ ， $y(x_0) = y_0$  的二步法

$$y_{n+1} = \frac{1}{2}(y_n + y_{n-1}) + \frac{h}{4}(4f_{n+1} - f_n + 3f_{n-1}) \quad (f_i = f(x_i, y_i))$$

是二阶的, 并求其局部截断误差主项。

9.11 解初值问题  $y' = f(x, y)$ ,  $y(x_0) = y_0$  用显式二步法  $y_{n+1} = a_0 y_n + a_1 y_{n-1} + h(\beta_0 f_n + \beta_1 f_{n-1})$ , 式中  $f_n = f(x_n, y_n)$ ,  $f_{n-1} = f(x_{n-1}, y_{n-1})$ 。试确定参数  $\alpha_0, \alpha_1, \beta_0, \beta_1$  使方法阶数尽可能高, 并求局部截断误差。

9.12 证明: 中点公式  $y_{n+1} = y_n + hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1)$ ,  $k_1 = f(x_n, y_n)$  是二阶的, 并求其局部截断误差主项。

9.13 证明: 下列公式对于任意参数  $t$  都是二阶的

$$\begin{cases} y_{n+1} = y_n + \frac{h}{2}(K_2 + K_3) \\ K_1 = f(x_n, y_n) \\ K_2 = f(x_n + th, y_n + thK_1) \\ K_3 = f(x_n + (1-t)h, y_n + (1-t)hK_2) \end{cases}$$

9.14 解初值问题  $y'(x) = 20(x - y)$ ,  $y(0) = 1$ , 为保证计算稳定性, 若用经典的四阶龙格-库塔方法, 步长  $0 < h < \underline{\hspace{1cm}}$ ; 若采用欧拉方法, 步长  $h$  的限制范围为  $\underline{\hspace{1cm}}$ ; 若采用隐式欧拉方法, 步长  $h$  范围为  $\underline{\hspace{1cm}}$ ; 若采用梯形公式, 步长  $h$  的范围为  $\underline{\hspace{1cm}}$ 。

9.15 解初值问题  $y' = f(x, y)$ ,  $y(x_0) = y_0$  的隐式欧拉公式  $y_{n+1} = y_n + hf(x_{n+1}, y_{n+1})$  是  $\underline{\hspace{1cm}}$  阶方法, 梯形法  $y_{n+1} = y_n + \frac{h}{2}[f_n + f(x_{n+1}, y_{n+1})]$  是  $\underline{\hspace{1cm}}$  阶方法。

9.16 解初值问题  $y' = -50(y + x)$ ,  $y(0) = 1$ , 若用四阶经典龙格-库塔方法, 步长  $h < \underline{\hspace{1cm}}$ 。若用隐式欧拉方法, 步长  $0 < h < \underline{\hspace{1cm}}$ 。

9.17 解初值问题  $y' = 10y - \frac{x}{y}$ ,  $1 \leq x \leq 2$ , 若用梯形法求解, 要使迭代法  $y_{n+1}^{(s+1)} = y_n + \frac{h}{2}[f_n + f(x_{n+1}, y_{n+1}^{(s)})]$  ( $s = 0, 1, \dots$ ) 收敛, 则步长  $h < \underline{\hspace{1cm}}$  ?

9.18 解初值问题  $y'(x) = f(x, y)$ ,  $y(x_0) = y_0$  的欧拉公式, 其局部截断误差主项是  $\underline{\hspace{1cm}}$ 。

## 第 10 章 矩阵特征值计算



### 学习要点

本章介绍矩阵特征值的估计和常用的数值计算方法,主要内容有:

- (1) 估计特征值范围的圆盘定理。
- (2) 计算实矩阵  $A$  的主特征值的幂法及反幂法。
- (3) 基于反射变换的求解矩阵的全部特征值的 QR 方法。
- (4) 基于平面旋转变换的求实对称矩阵  $A$  的全部特征值的雅可比方法。



### 教学建议

本章为选学内容。主要包括计算实矩阵  $A$  的主特征值的幂法和反幂法、求解矩阵的全部特征值与特征向量的 QR 方法以及求实对称矩阵  $A$  的全部特征值的雅可比方法。讲解本章全部内容,建议学时 8~10 学时。

## 10.1 引言

在许多科学和工程的计算问题中,经常要计算矩阵的特征值和特征向量。当矩阵的阶数较低时,可以用 2.3.6 节中的知识和方法求出矩阵的特征值和特征向量。但是在实际计算时,矩阵的阶数一般很高,用求解矩阵特征方程  $\det(\lambda I - A) = 0$  的方法计算矩阵的特征值是非常困难的。本章介绍用数值计算的方法求解矩阵的特征值和特征向量,主要包括幂法和反幂法、QR 方法以及雅可比方法。这些方法在实际中都是比较有效的方法。

在 2.3.6 节中,我们介绍了矩阵特征值的许多性质,在介绍求解矩阵特征值的数值计算方法之前,本节先介绍对矩阵特征值的上下界的估计。

**定义 10.1.1 (Gerschgorin 圆盘)** 设  $A = (a_{ij})_{n \times n}$ , 令

$$\textcircled{1} \quad r_i = \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}| \quad (i=1, 2, \dots, n) \quad (10.1.1)$$

$$\textcircled{2} \quad D_i = \{z \mid |z - a_{ii}| \leq r_i, z \in C\} \quad (i=1, 2, \dots, n) \quad (10.1.2)$$

称在复平面上以  $a_{ii}$  为圆心, 以  $r_i$  为半径的圆盘  $D_i$  为  $A$  的 Gerschgorin 圆盘。

**定理 10.1.1 (Gerschgorin 圆盘定理)** 设  $A = (a_{ij})_{n \times n}$ , 则

①  $A$  的每个特征值必属于某个圆盘  $D_i$  ( $i=1, 2, \dots, n$ ), 或者说,  $A$  的特征值都在复平面上  $n$  个圆盘的并集中。

② 如果  $A$  有  $m$  个圆盘组成一个连通的并集  $S$ , 且  $S$  与余下的  $n-m$  个圆盘是分离的, 则  $S$  内恰好包含  $A$  的  $m$  个特征值。特别地, 如果  $A$  的一个圆盘  $D_i$  是与其他圆盘分离的(即孤立的圆盘),

则  $D_i$  中精确地包含  $A$  的一个特征值。

**证明** 只证明结论①, 设  $\lambda$  为  $A$  的特征值, 即

$$Ax = \lambda x$$

式中  $x = (x_1, x_2, \dots, x_n)^T \neq 0$  为  $A$  的对应于  $\lambda$  的非 0 特征向量。

记  $|x_k| = \max_{1 \leq i \leq n} |x_i| = \|x\|_\infty \neq 0$ , 考虑  $Ax = \lambda x$  的第  $k$  个方程, 即

$$\sum_{j=1}^n a_{kj} x_j = \lambda x_k$$

或

$$(\lambda - a_{kk})x_k = \sum_{\substack{j=1 \\ j \neq k}}^n a_{kj} x_j$$

于是

$$|\lambda - a_{kk}| \cdot |x_k| \leq \sum_{j \neq k} |a_{kj}| \cdot |x_j| \leq |x_k| \sum_{j \neq k} |a_{kj}|$$

即

$$|\lambda - a_{ii}| \leq \sum_{j \neq i} |a_{ij}| = r_i$$

证毕。

**例 10.1.1** 已知  $A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 2 \end{pmatrix}$ , 求  $A$  的特征值和特征向量。

**解**  $A$  的特征方程为

$$\det(\lambda I - A) = \begin{vmatrix} \lambda - 2 & -1 & 0 \\ -1 & \lambda - 3 & -1 \\ 0 & -1 & \lambda - 2 \end{vmatrix} = 0$$

解得:  $\lambda_1 = 1, \lambda_2 = 2, \lambda_3 = 4$ 。

对应于  $\lambda$  的  $A$  的特征向量:  $x_1 = (1, -1, 1)^T, x_2 = (1, 0, -1)^T, x_3 = (1, 2, 1)^T$

**例 10.1.2** 估计矩阵  $A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 2 \end{pmatrix}$  的特征值的范围。

**解**  $D_1: |\lambda - 2| \leq 1$ , 即  $1 \leq \lambda \leq 3$

$D_2: |\lambda - 3| \leq 2$ , 即  $1 \leq \lambda \leq 5$

$D_3: |\lambda - 2| \leq 1$ , 即  $1 \leq \lambda \leq 3$

与例 10.1.1 的结论相符。

**定义 10.1.2 (瑞利商)** 设  $A \in R^{n \times n}, A = A^T$ , 对于任一非零向量  $x$ , 称

$$R(x) = \frac{(Ax, x)}{(x, x)}$$

为对应于向量  $x$  的瑞利 (Rayleigh) 商。

**定理 10.1.2 (瑞利商定理)** 设  $A \in R^{n \times n}, A = A^T$  (其特征值次序记为  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ )

则  $\forall x \neq 0, x \in R^n$ , 有

$$\textcircled{1} \lambda_n \leq \frac{(Ax, x)}{(x, x)} \leq \lambda_1$$



$$\textcircled{2} \quad \lambda_1 = \max \frac{(Ax, x)}{(x, x)}$$

$$\textcircled{3} \quad \lambda_n = \min \frac{(Ax, x)}{(x, x)}$$

**证明** 由于  $A$  为对称矩阵, 可将  $\lambda_1, \lambda_2, \dots, \lambda_n$  对应的特征向量  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$  正交规范化 (参见 2.3.7 节), 则有  $(\mathbf{x}_i, \mathbf{x}_j) = \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$ 。

设  $\mathbf{x} \neq 0$  为  $R^n$  的任一向量, 则有展开式

$$\mathbf{x} = \sum_{i=1}^n \alpha_i \mathbf{x}_i, \quad \|\mathbf{x}\|_2 = \left( \sum_{i=1}^n \alpha_i^2 \right)^{\frac{1}{2}} \neq 0$$

于是

$$\frac{(Ax, x)}{(x, x)} = \frac{\sum_{i=1}^n \alpha_i^2 \lambda_i}{\sum_{i=1}^n \alpha_i^2} \leq \frac{\lambda_1 \sum_{i=1}^n \alpha_i^2}{\sum_{i=1}^n \alpha_i^2} = \lambda_1$$

同理可证明  $\lambda_n \leq \frac{(Ax, x)}{(x, x)}$ , 从而定理 10.1.2 中的①成立。

在瑞利商中, 分别取  $\mathbf{x} = \mathbf{x}_1, \mathbf{x} = \mathbf{x}_n$ , 可达到瑞利商的最大值  $\lambda_1$  和最小值  $\lambda_n$ , 从而有:

$$\lambda_1 = \max \frac{(Ax, x)}{(x, x)}, \quad \lambda_n = \min \frac{(Ax, x)}{(x, x)}$$

证毕。

## 10.2 幂法及反幂法

### 10.2.1 幂法

幂法是一种计算实矩阵  $A$  的主特征值 (按模最大的特征值) 及其对应的特征向量的方法。此方法比较适合于大型稀疏矩阵。

设  $A$  有  $n$  个线性无关的特征向量  $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}$ ,  $A\mathbf{x}^{(i)} = \lambda_i \mathbf{x}^{(i)}$  ( $i=1, 2, \dots, n$ ) 且  $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|$ 。下面介绍用幂法求  $\lambda_1$  和  $\mathbf{x}^{(1)}$  的基本思想。

任取初始向量  $\mathbf{v}_0 \in R^n$ , 则  $\mathbf{v}_0$  可表示为:  $\mathbf{v}_0 = \alpha_1 \mathbf{x}^{(1)} + \alpha_2 \mathbf{x}^{(2)} + \dots + \alpha_n \mathbf{x}^{(n)}$  ( $\alpha_1 \neq 0$ )

则可构造

$$\begin{aligned} \mathbf{v}_1 &= A\mathbf{v}_0 = \alpha_1 \lambda_1 \mathbf{x}^{(1)} + \alpha_2 \lambda_2 \mathbf{x}^{(2)} + \dots + \alpha_n \lambda_n \mathbf{x}^{(n)} \\ \mathbf{v}_2 &= A\mathbf{v}_1 = \alpha_1 \lambda_1^2 \mathbf{x}^{(1)} + \alpha_2 \lambda_2^2 \mathbf{x}^{(2)} + \dots + \alpha_n \lambda_n^2 \mathbf{x}^{(n)} \end{aligned}$$

一般地, 有  $\mathbf{v}_k = A\mathbf{v}_{k-1} = \alpha_1 \lambda_1^k \mathbf{x}^{(1)} + \alpha_2 \lambda_2^k \mathbf{x}^{(2)} + \dots + \alpha_n \lambda_n^k \mathbf{x}^{(n)}$

$$= \lambda_1^k \left[ \alpha_1 \mathbf{x}^{(1)} + \alpha_2 \left( \frac{\lambda_2}{\lambda_1} \right)^k \mathbf{x}^{(2)} + \dots + \alpha_n \left( \frac{\lambda_n}{\lambda_1} \right)^k \mathbf{x}^{(n)} \right]$$

当  $k$  足够大时, 由于  $\left| \frac{\lambda_i}{\lambda_1} \right| < 1$  ( $i=2, 3, \dots, n$ ), 故  $\mathbf{v}_k \approx \lambda_1^k \alpha_1 \mathbf{x}^{(1)}$ ,  $\mathbf{v}_{k+1} \approx \lambda_1^{k+1} \alpha_1 \mathbf{x}^{(1)} \approx \lambda_1 \lambda_1^k \alpha_1 \mathbf{x}^{(1)} = \lambda_1 \mathbf{v}_k$ 。

而  $\mathbf{v}_{k+1} = A\mathbf{v}_k$ , 所以  $A\mathbf{v}_k \approx \lambda_1 \mathbf{v}_k$ 。因此,  $\mathbf{v}_k$  可以近似地作为对应于  $\lambda_1$  的特征向量。

如果用  $(\mathbf{v}_k)_i$  表示  $\mathbf{v}_k$  的第  $i$  个分量。则

$$\frac{(\mathbf{v}_{k+1})_i}{(\mathbf{v}_k)_i} \approx \frac{\lambda_1^{k+1} \alpha_1 (\mathbf{x}^{(1)})_i}{\lambda_1^k \alpha_1 (\mathbf{x}^{(1)})_i} = \lambda_1 \quad (i=1, 2, \dots, n) \quad (10.2.1)$$

这说明两个相邻迭代向量分量的比值收敛到主特征值  $\lambda_1$ 。

这种由非零初始向量  $\mathbf{v}_0$  及矩阵  $\mathbf{A}$  的幂  $\mathbf{A}^k$  构造向量序列  $\{\mathbf{v}_k\}$ ，用来计算  $\mathbf{A}$  的主特征值  $\lambda_1$  及相应的特征向量的方法称为幂法。

幂法的收敛速度由比值  $r = \left| \frac{\lambda_2}{\lambda_1} \right|$  来确定， $r$  越小，收敛速度就越快。但当  $r \approx \left| \frac{\lambda_2}{\lambda_1} \right| \approx 1$  时，收敛速度较慢。

应用幂法计算  $\mathbf{A}$  的主特征值  $\lambda_1$  及对应的特征向量时，如果  $|\lambda_i| > 1$ （或  $|\lambda_i| < 1$ ），则迭代向量  $\mathbf{v}_k = \lambda_1^k \alpha_1 \mathbf{x}^{(1)}$  将随  $k \rightarrow \infty$  而趋向于无穷（或 0），这样，在计算机实现时，就可能“溢出”，为了克服这个缺点，需将  $\mathbf{v}_k$  加以规范化，即令

$$\mathbf{u}_k = \frac{\mathbf{v}_k}{\max(\mathbf{v}_k)} \quad (k=0, 1, 2, \dots) \quad (10.2.2)$$

式中， $\max(\mathbf{v}_k)$  表示  $\mathbf{v}_k$  中取绝对值最大的分量，通常取  $\mathbf{v}_0 = \mathbf{u}_0 = (1, 1, \dots, 1)^T$ ，则

$$\begin{aligned} \mathbf{v}_1 &= \mathbf{A}\mathbf{v}_0 = \mathbf{A}\mathbf{u}_0 & \mathbf{u}_1 &= \frac{\mathbf{v}_1}{\max(\mathbf{v}_1)} = \frac{\mathbf{A}\mathbf{u}_0}{\max(\mathbf{A}\mathbf{u}_0)} \\ \mathbf{v}_2 &= \mathbf{A}\mathbf{u}_1 = \frac{\mathbf{A}^2 \mathbf{u}_0}{\max(\mathbf{A}\mathbf{u}_0)} & \mathbf{u}_2 &= \frac{\mathbf{v}_2}{\max(\mathbf{v}_2)} = \frac{\mathbf{A}^2 \mathbf{u}_0}{\max(\mathbf{A}^2 \mathbf{u}_0)} \\ &\dots & & \dots \\ \mathbf{v}_k &= \mathbf{A}\mathbf{u}_{k-1} = \frac{\mathbf{A}^k \mathbf{u}_0}{\max(\mathbf{A}^{k-1} \mathbf{u}_0)} & \mathbf{u}_k &= \frac{\mathbf{v}_k}{\max(\mathbf{v}_k)} = \frac{\mathbf{A}^k \mathbf{u}_0}{\max(\mathbf{A}^k \mathbf{u}_0)} \end{aligned}$$

由于  $\mathbf{v}_0 = \alpha_1 \mathbf{x}^{(1)} + \alpha_2 \mathbf{x}^{(2)} + \dots + \alpha_n \mathbf{x}^{(n)}$ ，因此

$$\begin{aligned} \mathbf{A}^k \mathbf{v}_0 &= \sum_{i=1}^n \alpha_i \lambda_i^k \mathbf{x}^{(i)} = \lambda_1^k \left[ \alpha_1 \mathbf{x}^{(1)} + \sum_{i=2}^n \alpha_i \left( \frac{\lambda_i}{\lambda_1} \right)^k \mathbf{x}^{(i)} \right] \\ \mathbf{u}_k &= \frac{\mathbf{A}^k \mathbf{v}_0}{\max(\mathbf{A}^k \mathbf{v}_0)} = \frac{\lambda_1^k \left[ \alpha_1 \mathbf{x}^{(1)} + \sum_{i=2}^n \alpha_i \left( \frac{\lambda_i}{\lambda_1} \right)^k \mathbf{x}^{(i)} \right]}{\max \left[ \lambda_1^k \left[ \alpha_1 \mathbf{x}^{(1)} + \sum_{i=2}^n \alpha_i \left( \frac{\lambda_i}{\lambda_1} \right)^k \mathbf{x}^{(i)} \right] \right]} \\ &= \frac{\alpha_1 \mathbf{x}^{(1)} + \sum_{i=2}^n \alpha_i \left( \frac{\lambda_i}{\lambda_1} \right)^k \mathbf{x}^{(i)}}{\max \left[ \alpha_1 \mathbf{x}^{(1)} + \sum_{i=2}^n \alpha_i \left( \frac{\lambda_i}{\lambda_1} \right)^k \mathbf{x}^{(i)} \right]} \rightarrow \frac{\mathbf{x}^{(1)}}{\max(\mathbf{x}^{(1)})} \quad (k \rightarrow \infty) \end{aligned}$$

这说明规范化向量序列  $\mathbf{u}_k = \frac{\mathbf{v}_k}{\max(\mathbf{v}_k)}$  ( $k=0, 1, 2, \dots$ ) 收敛到主特征值对应的特征向量。

同理可得

$$\mathbf{v}_k = \mathbf{A}\mathbf{u}_{k-1} = \frac{\mathbf{A}^k \mathbf{u}_0}{\max(\mathbf{A}^{k-1} \mathbf{u}_0)} = \frac{\lambda_1^k \left[ \alpha_1 \mathbf{x}^{(1)} + \sum_{i=2}^n \alpha_i \left( \frac{\lambda_i}{\lambda_1} \right)^k \mathbf{x}^{(i)} \right]}{\max \left( \lambda_1^{k-1} \left[ \alpha_1 \mathbf{x}^{(1)} + \sum_{i=2}^n \alpha_i \left( \frac{\lambda_i}{\lambda_1} \right)^{k-1} \mathbf{x}^{(i)} \right] \right)}$$

所以

$$\mu_k = \max(\mathbf{v}_k) = \frac{\lambda_1 \max \left( \alpha_1 \mathbf{x}^{(1)} + \sum_{i=2}^n \alpha_i \left( \frac{\lambda_i}{\lambda_1} \right)^k \mathbf{x}^{(i)} \right)}{\max \left( \alpha_1 \mathbf{x}^{(1)} + \sum_{i=2}^n \alpha_i \left( \frac{\lambda_i}{\lambda_1} \right)^{k-1} \mathbf{x}^{(i)} \right)} \rightarrow \lambda_1 \quad (k \rightarrow \infty)$$

下面给出幂法的算法框图（见图 10.2.1），算法中  $\mathbf{v}_0$  为初值， $\varepsilon$  为精度， $N$  为最大迭代次数。

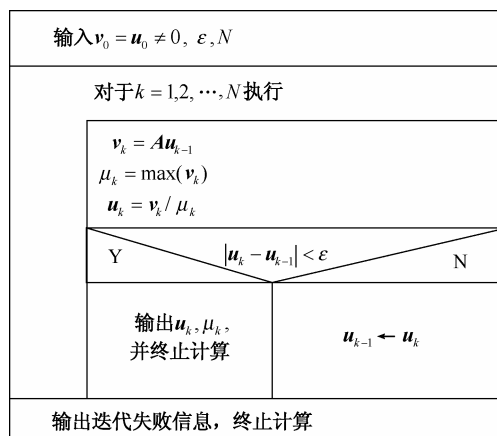


图 10.2.1 幂法的算法框图

例 10.2.1 用幂法求矩阵  $\mathbf{A} = \begin{pmatrix} 3 & 2 \\ 4 & 5 \end{pmatrix}$  的主特征值及对应的特征向量， $\varepsilon = 0.0001$ 。

解 取  $\mathbf{v}_0 = \mathbf{u}_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

$$\mathbf{v}_1 = \mathbf{A}\mathbf{u}_0 = \begin{pmatrix} 5 \\ 9 \end{pmatrix}, \quad \mu_1 = 9, \quad \mathbf{u}_1 = \begin{pmatrix} 0.5556 \\ 1.0000 \end{pmatrix}$$

$$\mathbf{v}_2 = \mathbf{A}\mathbf{u}_1 = \begin{pmatrix} 3.6667 \\ 7.2222 \end{pmatrix}, \quad \mu_2 = 7.2222, \quad \mathbf{u}_2 = \begin{pmatrix} 0.5077 \\ 1.0000 \end{pmatrix}$$

$$\mathbf{v}_3 = \mathbf{A}\mathbf{u}_2 = \begin{pmatrix} 3.5231 \\ 7.0308 \end{pmatrix}, \quad \mu_3 = 7.0308, \quad \mathbf{u}_3 = \begin{pmatrix} 0.5011 \\ 1.0000 \end{pmatrix}$$

$$\mathbf{v}_4 = \mathbf{A}\mathbf{u}_3 = \begin{pmatrix} 3.5033 \\ 7.0006 \end{pmatrix}, \quad \mu_4 = 7.0006, \quad \mathbf{u}_4 = \begin{pmatrix} 0.5000 \\ 1.0000 \end{pmatrix}$$

$$\mathbf{v}_5 = \mathbf{A}\mathbf{u}_4 = \begin{pmatrix} 3.5001 \\ 7.0000 \end{pmatrix}, \quad \mu_5 = 7.0000, \quad \mathbf{u}_5 = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix}$$

$$\mathbf{v}_6 = \mathbf{A}\mathbf{u}_5 = \begin{pmatrix} 3.5001 \\ 7.0000 \end{pmatrix}, \quad \mu_6 = 7.0000, \quad \mathbf{u}_6 = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix}$$

所以  $\lambda_1=7$ ，对应的特征向量为  $\begin{pmatrix} 0.5 \\ 1 \end{pmatrix}$ 。

当求  $A$  的主特征值  $\lambda_1$  时，其收敛速度主要由  $\frac{\lambda_2}{\lambda_1}=r$  来决定。当  $r \approx 1$  时，收敛速度较慢。此时，可引进矩阵  $B=A-pI$  ( $p$  为选择参数)，设  $A$  的特征值为  $\lambda_1, \lambda_2, \dots, \lambda_n$ ，则  $B$  的相应特征值为  $\lambda_1-p, \lambda_2-p, \dots, \lambda_n-p$ ，且  $A, B$  的特征向量相同。若求  $\lambda_1$ ，则适当选择  $p$ ，使  $\lambda_1-p$  仍为  $B$  的主特征值，且  $\left| \frac{\lambda_2-p}{\lambda_1-p} \right| < \left| \frac{\lambda_2}{\lambda_1} \right|$ ，对  $B$  应用幂法，求  $\lambda_1-p$  可加速，此法称为原点平移法。但选择适当的参数  $p$  有困难。

对于对称矩阵，可以用瑞利商进行加速，设  $A \in R^{n \times n}$  为对称矩阵，特征值满足： $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|$ ，且对应的特征值向量满足  $(x^{(i)}, x^{(j)}) = \delta_{ij}$ ，则用式 (10.2.2) 计算出的规范化向量  $u_k$  满足

$$\frac{(Au_k, u_k)}{(u_k, u_k)} = \lambda_1 + O\left(\left(\frac{\lambda_2}{\lambda_1}\right)^{2k}\right) \quad (10.2.3)$$

事实上，由于

$$u_k = \frac{v_k}{\max(v_k)} = \frac{A^k u_0}{\max(A^k u_0)}, \quad v_{k+1} = Au_k = \frac{A^{k+1} u_0}{\max(A^k u_0)}$$

因此

$$\frac{(Au_k, u_k)}{(u_k, u_k)} = \frac{(A^{k+1} u_0, u_0)}{(A^k u_0, u_0)} = \frac{\sum_{i=1}^n \alpha_i^2 \lambda_i^{2k+1}}{\sum_{i=1}^n \alpha_i^2 \lambda_i^{2k}} = \lambda_1 + O\left(\left(\frac{\lambda_2}{\lambda_1}\right)^{2k}\right)$$

这说明式 (10.2.3) 成立。

## 10.2.2 反幂法

反幂法是用于计算矩阵按模最小的特征值及其特征向量的方法，它也可以用来计算对应于一个给定近似特征值的特征向量。

设  $A \in R^{n \times n}$  为非奇异矩阵， $A$  的特征值为： $\lambda_1, \lambda_2, \dots, \lambda_n$ ，并且

$$|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$$

其相应的特征向量为  $x^{(1)}, x^{(2)}, \dots, x^{(n)}$  ( $i=1, 2, \dots, n$ )，则  $A^{-1}$  的特征值为： $\frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \dots, \frac{1}{\lambda_n}$ ，并且

$$\left| \frac{1}{\lambda_1} \right| \leq \left| \frac{1}{\lambda_2} \right| \leq \dots \leq \left| \frac{1}{\lambda_n} \right|$$

其对应的特征向量仍为  $x^{(1)}, x^{(2)}, \dots, x^{(n)}$ ，因此，计算  $A$  的按模最小的特征值  $\lambda_n$  的问题就是计算  $A^{-1}$  的按模最大的特征值  $\frac{1}{\lambda_n}$  的问题。

对于  $A^{-1}$  应用幂法（称为反幂法），可求得矩阵  $A^{-1}$  的主特征值  $\frac{1}{\lambda_n}$ ，从而求得  $A$  的按模最小的特征值  $\lambda_n$ 。

基本思想：任取  $v_0 = u_0 \neq 0$ ，构造向量序列如下

$$\begin{cases} \mathbf{v}_k = \mathbf{A}^{-1} \mathbf{u}_{k-1} \\ \mathbf{u}_k = \frac{\mathbf{v}_k}{\max(\mathbf{v}_k)} \end{cases} \quad (k=1, 2, \dots) \quad (10.2.4)$$

则  $\lim_{k \rightarrow \infty} \mathbf{u}_k = \frac{\mathbf{x}^{(n)}}{\max(\mathbf{x}^{(n)})}$ ,  $\lim_{k \rightarrow \infty} \max(\mathbf{v}_k) = \frac{1}{\lambda_n}$ , 收敛速度的比值为  $\left| \frac{\lambda_n}{\lambda_{n-1}} \right|$ 。

实际计算时, 可以对式 (10.2.4) 进行改进, 为避免矩阵求逆,  $\mathbf{v}_k = \mathbf{A}^{-1} \mathbf{u}_{k-1}$  可以通过求解线性方程组  $\mathbf{A} \mathbf{v}_k = \mathbf{u}_{k-1}$  得到。如果  $\mathbf{A}$  满足三角分解的条件, 可利用 4.3.2 节的知识求解。

反幂法也可以用原点平移法加速。如果  $p$  为矩阵  $\mathbf{A}$  的第  $i$  个特征值  $\lambda_i$  的近似值, 并且  $|\lambda_i - p| \ll |\lambda_j - p|$  ( $j=1, 2, \dots, n, j \neq i$ ), 于是矩阵  $\mathbf{B} = \mathbf{A} - p\mathbf{I}$  的特征值为:  $\lambda_j - p$  ( $j=1, 2, \dots, n$ )。 $\lambda_i - p$  是  $\mathbf{B}$  的按模最小的特征值, 可以对  $\mathbf{B}$  应用反幂法求出这个特征值及其对应的特征向量, 从而可得到  $\mathbf{A}$  的特征值及其对应的特征向量的近似值。

**例 10.2.2** 用反幂法求矩阵  $\mathbf{A} = \begin{pmatrix} -12 & 3 & 3 \\ 3 & 1 & -2 \\ 3 & -2 & 7 \end{pmatrix}$  的最接近 -13 的特征值及其特征向量。

**解** 令  $p = -13$ , 对  $\mathbf{B} = \mathbf{A} - p\mathbf{I}$  进行 LU 分解, 有

$$\mathbf{B} = \mathbf{A} - p\mathbf{I} = \begin{pmatrix} 1 & 3 & 3 \\ 3 & 14 & -2 \\ 3 & -2 & 20 \end{pmatrix} = \begin{pmatrix} 1 & & \\ 3 & 1 & \\ 3 & -\frac{11}{5} & 1 \end{pmatrix} \begin{pmatrix} 1 & 3 & 3 \\ & 5 & -11 \\ & & -\frac{66}{5} \end{pmatrix}$$

取  $\mathbf{v}_0 = \mathbf{u}_0 = (1, 1, 1)^T$ , 计算步骤和结果见式 (10.2.5) 和表 10.2.1。

$$\begin{cases} \mathbf{u}_k = \frac{\mathbf{v}_k}{\max(\mathbf{v}_k)} \\ L\mathbf{y}_k = \mathbf{u}_k \\ U\mathbf{v}_{k+1} = \mathbf{y}_k \end{cases} \quad (k=1, 2, \dots) \quad (10.2.5)$$

表 10.2.1 例 10.2.2 的计算结果

$k$	$\mathbf{v}_k$	$\mathbf{u}_k$	$\lambda_k$
0	(1, 1, 1)	(1, 1, 1)	
1	(-2.454 545 0, 6.666 666 9, 0.484 848 50)	(1, -0.271 604 96, -0.197 530 9)	-13.407 41
2	(-4.590 821 4, 1.078 189 4, 0.787 504 67)	(1, -0.234 537 7, -0.171 305 33)	-13.217 53
3	(-4.540 941 7, 1.067 640 5, 0.779 340 09)	(1, -0.235 114 35, -0.171 625 2)	-13.220 22
4	(-4.541 751 4, 1.067 790 0, 0.779 460 37)	(1, -0.231 053 5, -0.171 621 10)	-13.220 18
5	(-4.541 738 5, 1.067 787 7, 0.779 458 52)	(1, -0.235 105 48, -0.171 621 2)	-13.220 18

注: 其中  $\lambda_k = p + \frac{1}{\max(\mathbf{v}_{k+1})}$ 。

## 10.3 QR 方法

与幂法不同, QR 方法可以求解矩阵的全部特征值与特征向量。它是以矩阵的正交三角分解为基础的一种矩阵变换方法。

### 10.3.1 反射变换

**定义 10.3.1 (反射矩阵)** 形如  $H = I - 2uu^T$  的矩阵为反射矩阵或豪斯霍尔德(Householder)矩阵, 式中,  $u \in R^n$ , 且  $\|u\|_2 = 1$ ,  $I$  是  $n$  阶单位矩阵。

反射矩阵有如下性质:

(1)  $H = H^T$  ( $H$  是对称矩阵)

(2)  $H^T = H^{-1}$  ( $H$  是正交矩阵)

事实上,  $H^T H = (I - 2uu^T)(I - 2uu^T) = I - 2uu^T - 2uu^T + 4u(u^T u)u^T = I$ 。

(3)  $H^2 = I$  ( $H$  是对合矩阵)

(4)  $\det(H) = -1$

**定义 10.3.2 (反射变换)** 对于任意的非零向量  $x \in R^n$ , 称变换  $y = Hx$  为反射变换。

现在说明反射变换的几何意义, 以  $n=3$  为例 (三维空间), 设  $u$  为  $R^3$  上的一个单位向量,  $S$  为过原点且与  $u$  垂直的平面。对一切向量  $v \in R^3$ , 可以分解为  $v = v_1 + v_2$ , 其中  $v_1 \in S$ ,  $v_2 \perp S$ 。可以验证,  $u^T v_1 = 0$ ,  $v_2 = \alpha u$  ( $\alpha \in R$ )。于是

$$Hv_1 = v_1 - 2u(u^T v_1) = v_1, \quad Hv_2 = v_2 - 2u(u^T v_2) = v_2 - 2\alpha u = -v_2$$

最后得  $Hv = H(v_1 + v_2) = Hv_1 + Hv_2 = v_1 - v_2$ , 这样, 向量  $v$  经过反射变换得到的向量  $Hv$  是  $v$  关于平面  $S$  的对称向量。所以, 反射变换也称为镜面反射变换,  $H$  也称为初等反射矩阵, 如图 10.3.1 所示。

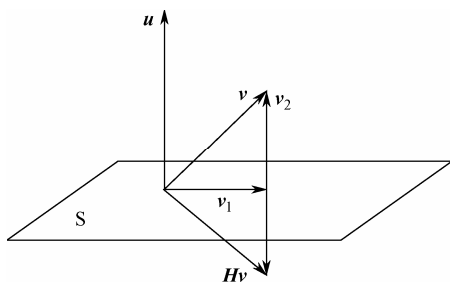


图 10.3.1 镜面反射变换

**定理 10.3.1 (保模变换)** 对于任意的非零向量  $x$ , 令  $y = Hx$ , 则  $\|x\|_2 = \|y\|_2$ 。

**证明**  $\|y\|_2 = \sqrt{(y, y)} = \sqrt{(Hx, Hx)} = \sqrt{(H^T Hx, x)} = \sqrt{(x, x)} = \|x\|_2$

**定理 10.3.2 (定理 10.3.1 的反命题)** 设  $x, y \in R^n$ ,  $x \neq y$ , 并且  $\|x\|_2 = \|y\|_2$ , 则存在  $n$  阶反射矩阵  $H$ , 使得  $y = Hx$ 。

**证明** 当  $x \neq y$  时, 可取  $u = \frac{(x-y)}{\|x-y\|_2}$ , 则  $u^T u = 1$ , 构造

$$H = I - 2 \frac{(x-y)(x-y)^T}{\|x-y\|_2^2}$$

$$\text{则 } Hx = Ix - 2 \frac{(x-y)(x-y)^T x}{\|x-y\|_2^2} = x - 2 \frac{(x-y)(x^T x - y^T x)}{\|x-y\|_2^2}$$

因为  $\|x\|_2 = \|y\|_2$ , 则

$$\|\mathbf{x} - \mathbf{y}\|_2^2 = (\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y}) = 2(\mathbf{x}^T \mathbf{x} - \mathbf{y}^T \mathbf{x})$$

所以  $\mathbf{H}\mathbf{x} = \mathbf{x} - (\mathbf{x} - \mathbf{y}) = \mathbf{y}$ 。证毕。

**例 10.3.1** 已知  $\mathbf{x} = (x_1, x_2, \dots, x_n)^T \in R^n$ ，且  $\mathbf{x} \neq \mathbf{0}$ ，求反射矩阵  $\mathbf{H}$ ，使得  $\mathbf{H}\mathbf{x} = \sigma \mathbf{e}_1$ ，式中  $\mathbf{e}_1 = (1, 0, 0, \dots, 0)^T$ ， $|\sigma| = \|\mathbf{x}\|_2$ 。

**解** 令  $\mathbf{u} = \frac{\mathbf{x} - \sigma \mathbf{e}_1}{\|\mathbf{x} - \sigma \mathbf{e}_1\|_2}$ ，取

$$\mathbf{H} = \mathbf{I} - 2 \frac{\mathbf{u}\mathbf{u}^T}{\|\mathbf{u}\|_2^2} \quad (10.3.1)$$

由于  $|\sigma| = \|\mathbf{x}\|_2$ ，则  $\sigma^2 = x_1^2 + x_2^2 + \dots + x_n^2$ 。

$$\begin{aligned} \|\mathbf{x} - \sigma \mathbf{e}_1\|_2 &= \sqrt{(x_1 - \sigma)^2 + x_2^2 + \dots + x_n^2} = \sqrt{x_1^2 - 2\sigma x_1 + \sigma^2 + x_2^2 + \dots + x_n^2} \\ &= \sqrt{\sigma^2 - 2\sigma x_1 + \sigma^2} = \sqrt{2\sigma(\sigma - x_1)} \end{aligned}$$

所以

$$\mathbf{u} = \frac{1}{\sqrt{2\sigma(\sigma - x_1)}} (x_1 - \sigma, x_2, x_3, \dots, x_n)$$

取

$$\mathbf{H} = \mathbf{I} - 2 \frac{(\mathbf{x} - \sigma \mathbf{e}_1)(\mathbf{x} - \sigma \mathbf{e}_1)^T}{\|\mathbf{x} - \sigma \mathbf{e}_1\|_2^2} = \mathbf{I} - \frac{(\mathbf{x} - \sigma \mathbf{e}_1)(\mathbf{x} - \sigma \mathbf{e}_1)^T}{\sigma(\sigma - x_1)}$$

则

$$\mathbf{H}\mathbf{x} = \mathbf{I}\mathbf{x} - \frac{(\mathbf{x} - \sigma \mathbf{e}_1)(\mathbf{x} - \sigma \mathbf{e}_1)^T \mathbf{x}}{\sigma(\sigma - x_1)} = \sigma \mathbf{e}_1$$

在实际计算时，为了防止  $\sigma$  和  $x_1$  相互抵消，可选取  $\sigma$  和  $x_1$  异号，即  $\sigma = -\text{sign}(x_1)\|\mathbf{x}\|_2$ 。

例如， $\mathbf{x} = (3, 5, 1, 1)^T \in R^4$ ，有  $\|\mathbf{x}\|_2^2 = 6$ ，取  $\sigma = -6$ ，按式 (10.3.1) 计算得

$$\mathbf{H} = \mathbf{I} - 2 \frac{\mathbf{u}\mathbf{u}^T}{\|\mathbf{u}\|_2^2} = \frac{1}{54} \begin{pmatrix} -27 & -45 & -9 & -9 \\ -45 & 29 & -5 & -5 \\ -9 & -5 & 53 & -1 \\ -9 & -5 & -1 & 53 \end{pmatrix}$$

可以直接验证  $\mathbf{H}\mathbf{x} = (-6, 0, 0, 0)^T$ 。

### 10.3.2 矩阵的 QR 分解

**定理 10.3.3** 设矩阵  $\mathbf{A} = (a_{ij})_{n \times n}$  非奇异，则存在正交三角分解  $\mathbf{A} = \mathbf{Q}\mathbf{R}$ ，其中  $\mathbf{Q}$  是  $n$  阶正交矩阵， $\mathbf{R}$  是非奇异的上三角矩阵。若限定  $\mathbf{R}$  的对角线元素均为正数，则此分解唯一。

**构造性证明** 设  $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)$ ， $\mathbf{a}_j = (a_{1j}, a_{2j}, \dots, a_{nj})^T$ ， $j = 1, 2, \dots, n$

第一步：对  $\mathbf{a}_1$  可确定  $\sigma_1$  和  $\mathbf{u}_1$  使  $\mathbf{H}_1 \mathbf{a}_1 = (\mathbf{I} - 2\mathbf{u}_1 \mathbf{u}_1^T) \mathbf{a}_1 = \sigma_1 \mathbf{e}_1$ 。

$$\text{令} \quad \mathbf{A}_2 = \mathbf{H}_1 \mathbf{A}_1 = \mathbf{H}_1 \mathbf{A} = \mathbf{H}_1 (\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n) = \begin{pmatrix} \sigma_1 & a_{12}^{(2)} & \cdots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{pmatrix}$$

式中,  $\mathbf{a}_2^{(2)} = \begin{pmatrix} a_{22}^{(2)} \\ \vdots \\ a_{n2}^{(2)} \end{pmatrix}$ , 且  $\mathbf{A}_2$  非奇异, 可知  $a_{22}^{(2)} \neq 0$ 。

第二步: 对  $\mathbf{a}_2^{(2)}$  可确定  $\sigma_2$  和  $\mathbf{u}_2$ , 构造

$$\overline{\mathbf{H}}_2 = (\mathbf{I} - 2\mathbf{u}_2\mathbf{u}_2^T), \quad \overline{\mathbf{H}}_2\mathbf{a}_2^{(2)} = \sigma_2\mathbf{e}_1^{(n-1)}, \quad \text{这里 } \mathbf{e}_1^{(n-1)} \text{ 表示 } n-1 \text{ 维单位向量。}$$

再令  $\mathbf{H}_2 = \begin{pmatrix} 1 & 0 \\ 0 & \overline{\mathbf{H}}_2 \end{pmatrix}$ , 则

$$\mathbf{A}_3 = \mathbf{H}_2\mathbf{A}_2 = \begin{pmatrix} \sigma_1 & a_{12}^{(2)} & \cdots & a_{1n}^{(2)} \\ 0 & \sigma_2 & \cdots & a_{2n}^{(3)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(3)} \end{pmatrix}$$

重复上述步骤, 经过  $n-1$  步, 得到初等反射阵  $\mathbf{H}_1, \mathbf{H}_2, \dots, \mathbf{H}_{n-1}$ , 使得

$$\mathbf{H}_{n-1}\mathbf{H}_{n-2}\cdots\mathbf{H}_2\mathbf{H}_1\mathbf{A} = \mathbf{R}$$

式中,  $\mathbf{R}$  为上三角矩阵, 也称为上 Hessenberg 矩阵。令

$$\mathbf{H}_{n-1}\mathbf{H}_{n-2}\cdots\mathbf{H}_2\mathbf{H}_1 = \mathbf{Q}^{-1}$$

这里  $\mathbf{Q}^{-1}$ ,  $\mathbf{Q}$  为正交矩阵, 则有  $\mathbf{Q}^{-1}\mathbf{A} = \mathbf{R}$ , 由此得到  $\mathbf{A} = \mathbf{QR}$ , 实现了  $\mathbf{A}$  的 QR 分解。证毕。

例 10.3.2 用反射变换, 将矩阵  $\mathbf{A} = \begin{pmatrix} 0 & 2 & 0 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{pmatrix}$  进行 QR 分解。

解 设  $\mathbf{A} = (a_1, a_2, a_3)$ , 求  $\mathbf{H}_1$  使  $\mathbf{H}_1\mathbf{a}_1 = \sigma_1\mathbf{e}_1$ , 其中  $\mathbf{e}_1 = (1, 0, 0)^T$

$$\sigma_1 = -\text{sign}(a_{11})\sqrt{\sum_{i=1}^3 a_{i1}^2} = \sqrt{0+2^2+0} = 2$$

$$\mathbf{u}_1 = \frac{\mathbf{a}_1 + \sigma_1\mathbf{e}_1}{\|\mathbf{a}_1 + \sigma_1\mathbf{e}_1\|_2} = \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, 0\right)^T$$

$$\mathbf{H}_1 = \mathbf{I} - 2\mathbf{u}_1\mathbf{u}_1^T = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{A}_2 = \mathbf{H}_1\mathbf{A}_1 = \begin{pmatrix} -2 & -1 & -2 \\ 0 & -2 & 0 \\ 0 & 2 & 1 \end{pmatrix}$$

同理得

$$\mathbf{H}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -0.707 & 0.707 \\ 0 & 0.707 & -0.707 \end{pmatrix}, \quad \mathbf{A}_3 = \mathbf{H}_2\mathbf{H}_1\mathbf{A}_1 = \begin{pmatrix} -2 & -1 & -2 \\ 0 & 2.828 & 0.707 \\ 0 & 0 & 0.707 \end{pmatrix} = \mathbf{R}$$

$$\mathbf{Q} = (\mathbf{H}_2\mathbf{H}_1)^{-1} = \begin{pmatrix} 0 & 0.707 & -0.707 \\ -1 & 0 & 0 \\ 0 & 0.707 & 0.707 \end{pmatrix}$$



### 10.3.3 QR 方法

本节介绍基于矩阵 QR 分解的求  $A$  的全部特征值的 QR 方法。

令  $A_1 = A$ ，对  $A_1$  进行 QR 分解

$$A_1 = Q_1 R_1$$

然后令  $A_2 = R_1 Q_1$ ，再对  $A_2$  进行 QR 分解

$$A_2 = Q_2 R_2$$

并令  $A_3 = R_2 Q_2$ ，这样继续下去，便得到了一个矩阵序列  $\{A_k\}$ ，即

$$\begin{cases} A_1 = A \\ A_k = Q_k R_k \\ A_{k+1} = R_k Q_k \end{cases} \quad (k=1, 2, \dots) \quad (10.3.2)$$

因为

$$\begin{aligned} A_{k+1} &= R_k Q_k = Q_k^{-1} A_k Q_k = Q_k^{-1} Q_{k-1}^{-1} A_{k-1} Q_{k-1} Q_k = \dots \\ &= Q_k^{-1} Q_{k-1}^{-1} \dots Q_2^{-1} Q_1^{-1} A Q_1 Q_2 Q_{k-1} \dots Q_k \end{aligned}$$

所以，式 (10.3.2) 中的矩阵序列  $\{A_k\}$ ，矩阵两两正交相似，即  $A_k$  与  $A$  有相同的特征值。而矩阵序列  $\{A_k\}$  本质上收敛于上三角矩阵或块上三角矩阵，且对角块为  $1 \times 1$  或  $2 \times 2$  矩阵。 $1 \times 1$  矩阵就是  $A$  的实特征值。每个  $2 \times 2$  矩阵含有  $A$  的一对复特征值。然后，可以用反幂法求对应的特征向量。QR 方法的算法框图如图 10.3.2 所示，图中  $\varepsilon, N$  分别为精度和最大迭代次数。

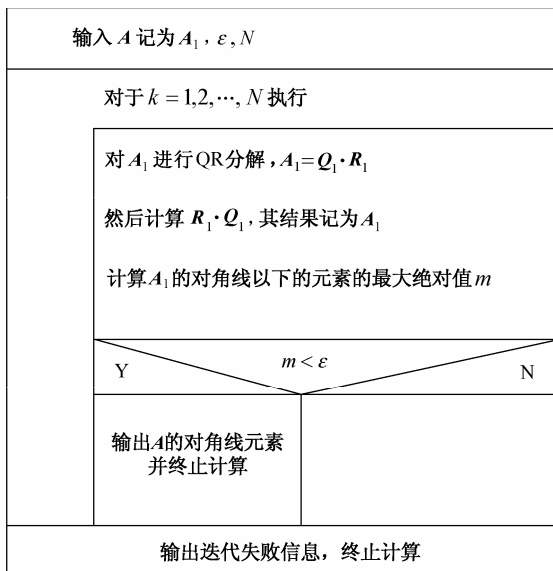


图 10.3.2 QR 方法的算法框图

例 10.3.3 用 QR 方法求矩阵  $A$  的特征值。

$$A = \begin{pmatrix} 0 & 2 & 0 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{pmatrix}$$

解 令  $A_1 = A$ ，对  $A_1$  进行 QR 分解

$$\mathbf{A}_1 = \mathbf{Q}_1 \mathbf{R}_1 = \begin{pmatrix} 0 & 0.707 & -0.707 \\ -1 & 0 & 0 \\ 0 & 0.707 & 0.707 \end{pmatrix} \begin{pmatrix} -2 & -1 & -2 \\ 0 & 2.828 & 0.707 \\ 0 & 0 & 0.707 \end{pmatrix}$$

然后令  $\mathbf{A}_2 = \mathbf{R}_1 \mathbf{Q}_1$ ，再对  $\mathbf{A}_2$  进行 QR 分解，一直进行下去，得到

$$\mathbf{A}_{14} = \begin{pmatrix} 3.6262 & 0.0056 & 0 \\ 0.0056 & -2.1413 & 0 \\ 0 & 0 & 0.5151 \end{pmatrix}$$

$$\mathbf{A}_{15} = \begin{pmatrix} 3.6262 & -0.0033 & 0 \\ -0.0033 & -2.1413 & 0 \\ 0 & 0 & 0.5151 \end{pmatrix}$$

所以， $\mathbf{A}$  的一个特征值为  $\lambda_1 = 3.6262, \lambda_2 = -2.1413, \lambda_3 = 0.5151$ 。

## 10.4 雅可比方法

雅可比 (Jacobi) 方法是求实对称矩阵  $\mathbf{A}$  的全部特征值和对应特征向量的方法，因为  $\mathbf{A}$  为实对称矩阵，由 2.3 节的相关知识可知， $\mathbf{A}$  可以对角化，即一定存在一个正交阵  $\mathbf{P}$ ，使  $\mathbf{P}^{-1} \mathbf{A} \mathbf{P} = \mathbf{D}$  (对角阵)， $\mathbf{D}$  的对角线上的元素即为  $\mathbf{A}$  的特征值。所以，雅可比方法是用一系列的正交相似变换，逐步构造出一个近似的正交矩阵  $\mathbf{P}$ ，使  $\mathbf{P}^{-1} \mathbf{A} \mathbf{P} = \mathbf{D}$ 。

### 10.4.1 平面旋转矩阵

我们先介绍将一个 2 阶实对称矩阵经过平面旋转变换化为对角矩阵的情况。

设  $\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ ，式中  $a_{12} = a_{21} \neq 0$

引入平面旋转矩阵  $\mathbf{P} = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix}$ ，因为  $\mathbf{P} \mathbf{P}^T = \mathbf{I}$ ，所以  $\mathbf{P}$  为正交矩阵。

选择适当的参数  $\theta$ ，使  $\mathbf{P}^{-1} \mathbf{A} \mathbf{P} = \mathbf{D} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} c_{11} & \\ & c_{22} \end{pmatrix}$

由矩阵的乘法可得

$$\begin{aligned} c_{11} &= a_{11} \cos^2 \theta + a_{22} \sin^2 \theta + a_{21} \sin 2\theta \\ c_{22} &= a_{11} \cos^2 \theta + a_{22} \sin^2 \theta - a_{21} \sin 2\theta \\ c_{12} &= c_{21} = \frac{1}{2}(a_{22} - a_{11}) \sin 2\theta + a_{21} \cos 2\theta \end{aligned}$$

若令  $c_{12} = c_{21} = 0$ ，则当  $a_{11} = a_{22}$  时，取  $\theta = \frac{\pi}{4}$  即可。

当  $a_{11} \neq a_{22}$  时，由于  $\tan 2\theta = \frac{2a_{12}}{a_{11} - a_{22}}$ ，可取  $\theta = \frac{1}{2} \arctan\left(\frac{2a_{12}}{a_{11} - a_{22}}\right)$ 。

例如，设  $\mathbf{A} = \begin{pmatrix} 2 & \frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & 1 \end{pmatrix}$ ，那么， $(x, y) \mathbf{A} \begin{pmatrix} x \\ y \end{pmatrix} = 1$  是平面上的一条椭圆曲线，写成多项式的形式

为:  $2x^2 + \sqrt{3}xy + y^2 = 1$ 。若将坐标轴逆时针旋转  $\theta = \frac{\pi}{6}$ , 则

$$P^{-1}AP = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} A \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \end{pmatrix} A \begin{pmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} \end{pmatrix} = \begin{pmatrix} \frac{5}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}$$

得到标准化的方程  $(x, y) \begin{pmatrix} \frac{5}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = 1$ , 写成多项式的形式为:  $5x^2 + y^2 = 2$ 。

现在将这种思想推广到  $n$  阶情况, 定义  $R^n$  中的平面旋转变换矩阵

$$R(p, q, \theta) = \begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \cos \theta & & \sin \theta & \\ & & & 1 & & \\ & & & & 1 & \\ & & -\sin \theta & & \cos \theta & \\ & & & & & 1 \\ & & & & & & 1 \end{pmatrix} \quad \begin{array}{l} \text{第 } p \text{ 行} \\ \\ \text{第 } q \text{ 行} \end{array} \quad (10.4.1)$$

矩阵  $R(p, q, \theta)$  具有如下性质:

- ①  $R$  为正交矩阵 (即  $RR^T = I$ )。
- ②  $R$  和单位矩阵  $I$  仅在  $(p, p), (p, q), (q, p), (q, q)$  这 4 个位置上的元素不同, 其他元素相同。
- ③ 因为  $A$  为实对称矩阵, 所以相似变换  $RAR^T = B$  是对称阵, 且  $B$  与  $A$  的特征值相同。
- ④  $RA$  只改变  $A$  的第  $p$  行与第  $q$  行元素,  $AR^T$  只改变  $A$  的第  $p$  列与第  $q$  列元素, 所以

$RAR^T$  只改变第  $p$  行, 第  $q$  行, 第  $p$  列和第  $q$  列的元素。

雅可比方法就是用一系列的旋转相似变换逐渐将  $A$  化为对角阵的过程

$$\begin{cases} A_0 = A \\ A_{k+1} = R_{k+1} A_k R_{k+1}^T \end{cases} \quad (k = 0, 1, 2, \dots) \quad (10.4.2)$$

恰当地选取  $R_{k+1}$ , 可使  $A_k$  趋于对角矩阵。设  $R_{k+1} = R(p, q, \theta)$ , 由于  $R_{k+1}$  为正交矩阵,  $A_{k+1}$  与  $A_k$  的差别仅在于第  $p$  行, 第  $q$  行, 第  $p$  列和第  $q$  列的元素不同。

由矩阵的乘法可得

$$\begin{cases} a_{pj}^{k+1} = a_{pj}^k \cos \theta + a_{qj}^k \sin \theta = a_{jp}^{k+1} \\ a_{qj}^{k+1} = -a_{pj}^k \sin \theta + a_{qj}^k \cos \theta = a_{jq}^{k+1} \end{cases} \quad (j \neq p, q) \quad (10.4.3)$$

$$\begin{cases} a_{pp}^{k+1} = a_{pp}^k \cos^2 \theta + 2a_{pq}^k \sin \theta \cos \theta + a_{qq}^k \sin^2 \theta \\ a_{qq}^{k+1} = a_{pp}^k \sin^2 \theta - 2a_{pq}^k \sin \theta \cos \theta + a_{qq}^k \cos^2 \theta \\ a_{pq}^{k+1} = \frac{1}{2}(a_{qq}^k - a_{pp}^k) \sin 2\theta + a_{pq}^k \cos 2\theta = a_{qp}^{k+1} \end{cases} \quad (10.4.4)$$

$$a_{ij}^{k+1} = a_{ij}^k \quad (i, j \neq p, q) \quad (10.4.5)$$

由式 (10.4.3) 知: 当  $j \neq p, q$  时,  $(a_{pj}^{k+1})^2 + (a_{qj}^{k+1})^2 = (a_{pj}^k)^2 + (a_{qj}^k)^2$

由式 (10.4.4) 知:  $(a_{pp}^{k+1})^2 + (a_{qq}^{k+1})^2 + 2(a_{pq}^{k+1})^2 = (a_{pp}^k)^2 + (a_{qq}^k)^2 + 2(a_{pq}^k)^2$

由式 (10.4.5) 知: 当  $i, j \neq p, q$  时,  $(a_{ij}^{k+1})^2 = (a_{ij}^k)^2$

由式 (10.4.4) 知, 要使  $a_{pq}^{k+1} = a_{qp}^{k+1} = 0$ , 可以选择合适的  $\theta$ , 使得

$$\tan 2\theta = \frac{2a_{pq}^k}{a_{pp}^k - a_{qq}^k}$$

现在, 引入记号

$$D(A) = \sum_{i=1}^n a_{ii}^2, \quad S(A) = \sum_{\substack{i,j=1 \\ i \neq j}}^n a_{ij}^2$$

因为  $a_{ii}^{k+1} = a_{ii}^k$  ( $i \neq p, q$ ), 所以

$$\begin{cases} D(A_{k+1}) = D(A_k) + 2(a_{pq}^k)^2 \\ S(A_{k+1}) = S(A_k) - 2(a_{pq}^k)^2 \end{cases} \quad (10.4.6)$$

则  $A_k$  按式 (10.4.2) 经旋转变换为  $A_{k+1}$  后, 对角线元素平方和增大, 非对角线元素平方和减小。因此这说明  $A_k$  将逐渐趋向于一个对角矩阵。

**定理 10.4.1** 设  $A$  为实对称矩阵, 则从  $A_0 = A$  出发, 经过式 (10.4.2) 计算产生的矩阵序列  $\{A_k\}$  收敛于对角阵  $D = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ , 且  $\lambda_1, \lambda_2, \dots, \lambda_n$  就是  $A$  的全部特征值, 而  $H_k = R_k \cdots R_1$  的行向量为对应的特征向量 (证明略)。

## 10.4.2 雅可比方法及其改进

根据定理 10.4.1, 可以得到计算实对称矩阵  $A$  的全部特征值的雅可比方法。下面给出雅可比方法的算法框图 (见图 10.4.1), 算法中  $A$ ,  $R = I$  为初值,  $\varepsilon$  为精度,  $N$  为最大迭代次数。

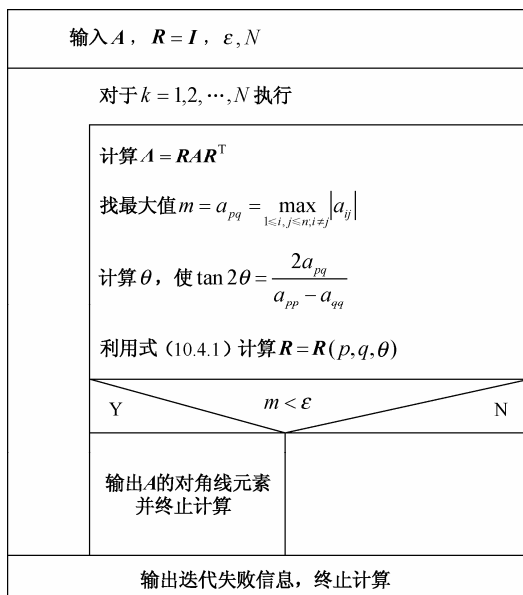


图 10.4.1 雅可比方法的算法框图

关于雅可比方法的几点说明:

① 在对实对称矩阵  $A$  施行一系列的平面旋转变换过程中, 前一步变为 0 的元素, 后一步可能又变为非零元素, 但整体趋势是把非对角线元素化为充分小。

② 在平面旋转变换过程中, 每次都必须重新计算一次  $\sin \theta$  和  $\cos \theta$ , 但可以不计算出  $\theta$ , 直接用三角函数之间的关系式表示即可。具体推导过程为

$$\tan 2\theta = \frac{2 \tan \theta}{1 - \tan^2 \theta} = \frac{2a_{ij}}{a_{ii} - a_{jj}} = \frac{1}{c}, \text{ 式中 } c = \frac{a_{ii} - a_{jj}}{2a_{ij}} \quad (a_{ij} \neq 0)$$

由  $1 - \tan^2 \theta = 2c \tan \theta$ ,  $\tan^2 \theta + 2c \tan \theta - 1 = 0$ , 求出  $\tan \theta = t$ , 则

$$\sin \theta = \frac{1}{\sqrt{1+t^2}}, \cos \theta = \frac{t}{\sqrt{1+t^2}}。$$

③ 找最大值  $m = a_{pq} = \max_{1 \leq i, j \leq n; i \neq j} |a_{ij}|$  比较费时间, 可以用雅可比过关法改进。先计算  $A$  的非对角线元素平方和  $S(A) = \sum_{\substack{i, j=1 \\ i \neq j}}^n a_{ij}^2$ , 记为  $v_0^2$ ,  $v_0^2 = 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^n a_{ij}^2$ , 将  $v_1 = \frac{v_0}{n}$  作为第一次过关数,

在上三角元素中按行  $(a_{12}, a_{13}, \dots, a_{1n}, a_{23}, \dots, a_{2n}, \dots, a_{n-1, n})$  扫描。只要  $|a_{ij}| \geq v_1$ , 就把  $a_{ij}$  作为主元素进行平面旋转变换; 若  $|a_{ij}| < v_1$ , 则让  $a_{ij}$  过关。一遍扫描结束后, 继续进行第二遍扫描, 原因是原来满足  $|a_{ij}| < v_1$  的过关元素, 经过其他遍变换后有可能变得不过关。直至某一遍扫描时, 全部元素都过关, 再设第二道关口,  $v_2 = \frac{v_1}{n}$ , 重复前面过程, 然后再设  $v_3, v_4, \dots$  等关口, 直到

$v_t \leq \left(\frac{\varepsilon}{n}\right) v_0$  为止。

例 10.4.1 用雅可比方法计算对称矩阵  $A$  的特征值和对应的一组特征向量。

$$A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$$

解 计算过程见表 10.4.1。

表 10.4.1 例 10.4.1 的计算过程

$k$	$A_k$	$\sin \theta_k, \cos \theta_k$	$A_k(p_k, q_k)$
0	$\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$	0.7071, 0.7071	$A_0(1, 2) = -1$
1	$\begin{pmatrix} 3 & 0 & 0.7071 \\ 0 & 1 & -0.7071 \\ 0.7071 & -0.7071 & 2 \end{pmatrix}$	0.4597, 0.8880	$A_1(1, 3) = 0.70101$
2	$\begin{pmatrix} 3.3660 & -0.3250 & 0 \\ -0.3250 & 1 & -0.6279 \\ 0 & -0.6279 & 1.6339 \end{pmatrix}$	0.5242, 0.8516	$A_2(2, 3) = -0.6279$
3	$\begin{pmatrix} 3.3600 & -0.1703 & 0.2768 \\ -0.1703 & 2.0204 & 0 \\ 0.2768 & 0 & 0.6135 \end{pmatrix}$	0.0990, 0.9950	$A_3(3, 1) = -0.2768$

续表

$k$	$A_k$	$\sin \theta_k, \cos \theta_k$	$A_k(p_k, q_k)$
4	$\begin{pmatrix} 3.3935 & -0.1695 & 0 \\ -0.1695 & 2.0204 & -0.0168 \\ 0 & -0.0168 & 0.5859 \end{pmatrix}$	0.1207, 0.9928	$A_4(2,1) = -0.1695$
5	$\begin{pmatrix} 3.4142 & 0 & 0.0020 \\ 0 & 1.9998 & -0.0167 \\ 0.0020 & -0.0167 & 0.5859 \end{pmatrix}$		

由表 10.4.1 可知, 计算迭代 5 次后, 特征值的近似值为

$$\lambda_1 \approx 3.4142, \lambda_2 \approx 1.9998, \lambda_3 \approx 0.5859$$

$$H_4 = R_4 R_3 R_2 R_1 = \begin{pmatrix} 0.5000 & 0.7071 & 0.5000 \\ -0.7071 & 0 & 0.7071 \\ 0.5000 & -0.7071 & 0.5000 \end{pmatrix}$$

由此可以得到与特征值  $\lambda_1 \approx 3.4142, \lambda_2 \approx 1.9998, \lambda_3 \approx 0.5859$  对应的特征向量为

$$\begin{pmatrix} 0.5000 \\ -0.7071 \\ 0.5000 \end{pmatrix}, \begin{pmatrix} 0.7071 \\ 0 \\ -0.7071 \end{pmatrix}, \begin{pmatrix} 0.5000 \\ 0.7071 \\ 0.5000 \end{pmatrix}$$

事实上,  $A$  的特征值和精确值为

$$\lambda_1 = 2 + \sqrt{2}, \lambda_2 = 2, \lambda_3 = 2 - \sqrt{2}$$

对应的一组特征向量为

$$\begin{pmatrix} \frac{1}{2} \\ \frac{\sqrt{2}}{2} \\ \frac{1}{2} \end{pmatrix}, \begin{pmatrix} \frac{\sqrt{2}}{2} \\ 0 \\ -\frac{\sqrt{2}}{2} \end{pmatrix}, \begin{pmatrix} \frac{1}{2} \\ \frac{\sqrt{2}}{2} \\ \frac{1}{2} \end{pmatrix}$$

## 本章小结

本章首先介绍用于估计矩阵特征值范围的圆盘定理和瑞利商定理, 然后重点介绍求矩阵特征值的幂法、QR 方法和雅可比方法。

幂法是一种计算实矩阵  $A$  的主特征值 (按模最大的特征值) 及其对应的特征向量的方法。反幂法用于计算矩阵按模最小的特征值及其特征向量的方法, 它也可以用来计算对应于一个给定近似特征值的特征向量。

QR 方法可以求解矩阵的全部特征值与特征向量。它是以矩阵的正交三角分解为基础的一种矩阵变换方法, 主要利用反射变换求正交矩阵  $Q$  和上三角矩阵  $R$ 。

雅可比 (Jacobi) 方法是求实对称矩阵  $A$  的全部特征值和对应特征向量的方法。雅可比方法用一系列的旋转相似变换逐渐将  $A$  化为对角阵, 即用一系列的正交相似交换, 逐步构造出一个近似的正交矩阵  $P$ , 使  $P^{-1}AP = D$ 。

## 习题 10

10.1 用幂法求矩阵  $A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{pmatrix}$  的主特征值及对应的特征向量。

10.2 取  $p = -0.5$ ，用原点平移幂法求第 1 题的主特征值及对应的特征向量。

10.3 用反幂法求矩阵  $A = \begin{pmatrix} 9 & 10 & 8 \\ 10 & 5 & -1 \\ 8 & -1 & 3 \end{pmatrix}$  的按模最小的特征值。

10.4 用带原点平移的反幂法求矩阵  $A = \begin{pmatrix} -12 & 3 & 3 \\ 3 & 1 & -2 \\ 3 & -2 & 7 \end{pmatrix}$  的最接近于  $p = -13$  的特征值及对应的特征向量。

10.5 用反射矩阵将向量  $\mathbf{x} = (2, 3, 0, 5)^T$  变为与单位向量  $\mathbf{e}_1 = (1, 0, 0, 0)^T$  平行的向量。

10.6 用反射矩阵对矩阵  $A = \begin{pmatrix} 0 & 2 & 0 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{pmatrix}$  进行 QR 分解。

10.7 用反射矩阵将对称矩阵  $A = \begin{pmatrix} 6 & 2 & 3 & 1 \\ 2 & 5 & 4 & 8 \\ 3 & 4 & 9 & 1 \\ 1 & 8 & 1 & 7 \end{pmatrix}$  化为三对角矩阵。

10.8 用 QR 方法计算矩阵  $A = \begin{pmatrix} 3 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{pmatrix}$  的全部特征值。

10.9 用带位移的 QR 方法计算  $A = \begin{pmatrix} 1 & 2 & 0 \\ 2 & -1 & 1 \\ 0 & 1 & 3 \end{pmatrix}$  的全部特征值。

10.10 用雅可比方法求实对称矩阵  $A = \begin{pmatrix} 1 & 2 & 0 \\ 2 & -1 & 1 \\ 0 & 1 & 3 \end{pmatrix}$  的全部特征值及对应的特征向量。

## 第 11 章 函数优化计算



### 学习要点

本章介绍求一元和多元函数最优解的条件及方法。

一元函数优化计算方法有：牛顿法、拟牛顿法和黄金分割法。

多元函数优化计算方法有：梯度法、牛顿法、共轭方向法和拟牛顿法。



### 教学建议

本章为选学内容，主要包括牛顿法、梯度法、共轭方向法和拟牛顿法。建议学时为 8~10 学时。

## 11.1 引言

函数优化计算所研究的问题是讨论在所有的解中什么样的解最优，以及如何求出最优解。这是一个古老的课题，早在 17 世纪，英国数学家牛顿（Newton）发明微积分时就提出函数极值问题。1847 年，法国数学家柯西（Cauchy）研究了函数沿着什么方向下降最快的问題。

首先从下面的几个例子说起。

**例 11.1.1（容积问题）** 设有  $8\text{cm} \times 5\text{cm}$  的矩形纸板一块，要将 4 角截去 4 个小方块，折叠成无盖纸盒。问小方块的大小是多少时，才能使纸盒的容积最大？

**解** 设小方块的边长为  $x$ ，则纸盒容积为

$$v = (8 - 2x)(5 - 2x)x = 4x^3 - 26x^2 + 40x$$

此问题的数学模型为：

$$\text{求极大值} \quad v = 4x^3 - 26x^2 + 40x \quad (0 \leq x \leq 2.5)$$

$$\text{或求极小值} \quad u = -(4x^3 - 26x^2 + 40x) \quad (0 \leq x \leq 2.5)$$

用解析方法计算：求  $u' = -(12x^2 - 52x + 40)$ ，令  $u' = 0$ ，得  $x_1 = 1, x_2 = 10/3$ 。再求  $u'' = -(24x - 52)$ ，由  $u''(1) > 0$  与  $u''(10/3) < 0$  可知  $x = 1$ ，为问题的解，即小方块边长为  $1\text{cm}$  时，纸盒容积最大为  $v = 18\text{cm}^3$ 。

**例 11.1.2（距离问题）** 已知两条曲线  $c_1: y_1 = 2\sin(x)$  与  $c_2: y_2 = x^2 + 2$  无交点。试求  $c_1$  与  $c_2$  之间的最短路径  $d$ 。

**解** 在  $c_1$  上取点  $P(x_1, y_1)$ ，在  $c_2$  上取  $Q(x_2, y_2)$ ，则

$$d = PQ = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$\text{或} \quad d^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$$

约束条件为：  $y_1 = 2\sin x_1$ ，  $y_2 = x_2^2 + 2$ 。若设  $y_1 = x_3, y_2 = x_4$ ，则此问题的数学模型为：



求极小值  $f = (x_1 - x_2)^2 + (x_3 - x_4)^2$

$$\text{约束条件为} \begin{cases} 2\sin x_1 - x_3 = 0 \\ x_2^2 + 2 - x_4 = 0 \end{cases}$$

此问题很难用解析方法计算求解，必须寻求数值计算方法求解。

一般地，函数优化问题常被写成如下形式

$$\begin{aligned} & \min f(\mathbf{x}) \\ \text{s.t.} \quad & \begin{cases} g_i(\mathbf{x}) \geq 0 & i = 1, 2, \dots, l \\ h_j(\mathbf{x}) = 0 & j = 1, 2, \dots, m \end{cases} \end{aligned} \quad (11.1.1)$$

式中，s.t. 是 subject to 的意思，表示约束条件。 $\mathbf{x} \in R^n$  是  $n$  维向量， $f(\mathbf{x}): R^n \rightarrow R$  称为目标函数， $g_i(\mathbf{x}): R^n \rightarrow R$  称为不等式约束函数， $h_j(\mathbf{x}): R^n \rightarrow R$  称为等式约束函数。

同时满足  $f, g, h$  的自变量集合称为上述优化问题的可行域。可行域内的任一点都为可行点。当只有目标函数  $f(\mathbf{x})$ ，而没有约束条件  $g_i(\mathbf{x})$ ， $h_j(\mathbf{x})$  时，此问题称为无约束优化问题。例 11.1.1 是无约束优化问题，例 11.1.2 是有约束优化问题。本章只介绍无约束优化问题。

**定义 11.1.1** 设无约束优化问题： $\min f(x)$ ， $\mathbf{x} \in R^n$ ，若  $\exists \mathbf{x}^* \in R^n$  使对  $\forall \mathbf{x} \in R^n$ ，总有  $f(\mathbf{x}^*) \leq f(\mathbf{x})$ ，则称  $\mathbf{x}^*$  为问题的全局最优解。若  $\exists \mathbf{x}^* \in R^n$ ，在  $\mathbf{x}^*$  的一个  $\varepsilon$  邻域： $N(\mathbf{x}^*, \varepsilon) = \{\mathbf{x} \mid \|\mathbf{x}^* - \mathbf{x}\| < \varepsilon\}$ ，使得对于  $\mathbf{x}^*$  与  $\mathbf{x} \in N(\mathbf{x}^*, \varepsilon)$ ，总有  $f(\mathbf{x}^*) \leq f(\mathbf{x})$ ，则称  $\mathbf{x}^*$  为问题的局部最优解。若将“ $\leq$ ”号改为“ $<$ ”，则为严格全局（局部）最优解。

## 11.2 一元函数优化计算

当  $n=1$ ， $x \in R^1$  时，式 (11.1.1) 变为一元函数优化问题：

$$\min f(x), \quad x \in R^1 \quad (11.2.1)$$

一元函数优化问题 (11.2.1) 有解的必要性条件是：若  $x^*$  是问题 (11.2.1) 的局部最优解，则  $x^*$  必使  $f'(x^*) = 0$ 。一元函数优化问题 (11.2.1) 有解的充分性条件是：若  $x^*$  使  $f'(x^*) = 0$ ，且  $f''(x^*) > 0$ ，则  $x^*$  是问题 (11.2.1) 的局部最优解。一元函数  $y = f(x)$  对应于一条平面曲线  $C$ ，必要性条件对应于在点  $x^*$  处曲线  $C$  的切线是水平的，或者说， $x^*$  是曲线  $C$  的一个驻点；充分性条件对应于在点  $x^*$  处，不仅曲线  $C$  的切线是水平的，同时曲线  $C$  还是（向下）凸的。

一元函数优化问题，一般不能利用导数为 0，求函数极值的方法（解析法）求解，必须要用迭代法求其数值解。本节介绍牛顿法、拟牛顿法和黄金分割法。

### 11.2.1 牛顿法

设有一元函数  $y = f(x)$ ，在  $(a, b)$  上  $f'(x)$  与  $f''(x)$  都存在且都连续，令  $x_0 \in (a, b)$ ，将  $f(x)$  在点  $x_0$  处二阶泰勒展开，有

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + O(|x - x_0|^2)$$

令

$$g(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2$$

有  $f(x) \approx g(x)$ ，可得原曲线  $y = f(x)$  在  $x_0$  点附近的一条近似抛物线，求

$$g'(x) = f'(x_0) + f''(x_0)(x - x_0)$$

令  $g'(x) = 0$ ，得  $x = x_0 - \frac{f'(x_0)}{f''(x_0)}$ ，取  $x_0$  为初始点，可得迭代公式

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} \quad (11.2.2)$$

称式 (11.2.2) 为求一元函数最优解的牛顿迭代公式，简称为一元牛顿迭代公式。其几何意义如图 11.2.1 所示。

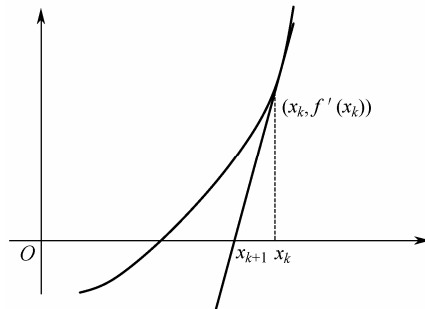


图 11.2.1 牛顿迭代公式的几何意义

一般地，若有二次抛物线(记为  $c_1$ )在点  $x_k$  处与曲线  $y = f(x)$  (记为  $c_2$ ) 具有相同高度  $g(x_k) = f(x_k)$ ，相同切线  $g'(x_k) = f'(x_k)$  及相同曲率  $g''(x_k) = f''(x_k)$ ，则称  $c_1$  为  $c_2$  在点  $x_k$  处的密切抛物线。由图 11.2.1 可以看出，一元牛顿迭代公式用  $x_k$  点的密切抛物线  $y = g(x)$  的切线段代替原曲线  $y = f(x)$  的弧线段。

如果给定一个初始点  $x_0$ ，按式 (11.2.2) 计算，就可得到一个点列  $\{x_k\}$ ，当计算到条件  $|f'(x_k)| < \varepsilon$  ( $\varepsilon > 0$ ，为计算精度) 满足时结束，将  $x_k$  作为要求解的一个近似值。其算法框图如图 11.2.2 所示。

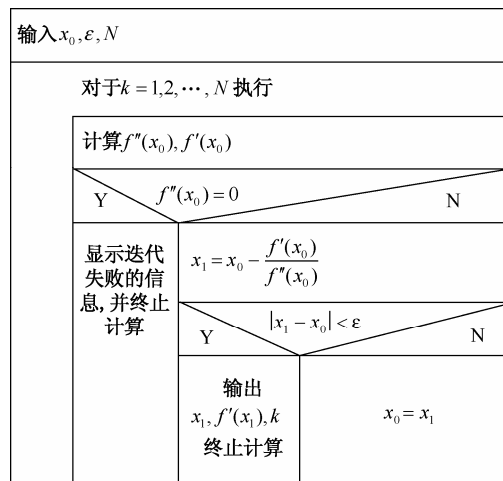


图 11.2.2 一元牛顿法的算法框图

例 11.2.1 用牛顿法求解问题  $\min f(x) = x^5 + x^3 - 7x$  的最优解。

**解** 由于  $f'(x) = 5x^4 + 3x^2 - 7$ ,  $f'(0) = -7$ ,  $f'(1) = 1$ , 且  $f'(x)$  在  $[0,1]$  内连续, 在  $[0,1]$  内必有驻点  $x^*$  使  $f'(x^*) = 0$ 。利用式 (11.2.2) 计算, 为此先求

$$f'(x) = 5x^4 + 3x^2 - 7, f''(x) = 20x^3 + 6x$$

取  $x_0 = 1$ , 有  $f'(x_0) = 1$ ,  $f''(x_0) = 26$

$$x_1 = x_0 - \frac{f'(x_0)}{f''(x_0)} = 1 - \frac{1}{26} = 0.961538 \approx 0.9615, f'(x_1) = 0.04678$$

$$x_2 = x_1 - \frac{f'(x_1)}{f''(x_1)} = \cdots = 0.959513, f'(x_2) = -0.00018$$

$$x_3 = \cdots = 0.959508, f'(x_3) = -0.00018$$

由于  $f'(x_3) \approx 0$ , 再由  $f''(x_3) = 23.424170 > 0$  可知,  $x^* \approx x_3 = 0.9595$  为最优解。

牛顿法特点如下。

优点: 收敛速度快, 为二阶局部收敛。

缺点: 需要计算二阶导数, 还需要一个较好的初始点。

## 11.2.2 拟牛顿法

作曲线  $y = f'(x)$  在  $[a,b]$  内的割线, 有

$$\frac{f'_b - f'_a}{b - a} = \frac{f'_b - 0}{b - x_1}, \quad b - x_1 = \frac{b - a}{f'_b - f'_a} \cdot f'_b$$

$$x_1 = b - \frac{b - a}{f'_b - f'_a} \cdot f'_b \quad \text{或} \quad x_1 = a - \frac{b - a}{f'_b - f'_a} \cdot f'_a$$

构造拟牛顿法的迭代公式

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f'_k - f'_{k-1}} \cdot f'_k \quad (11.2.3)$$

拟牛顿法 (割线法) 用弦线段代替弧线段, 其速度比切线法慢, 为 1.618 阶的超线性收敛, 但它只用一阶导数而不用二阶导数。

**例 11.2.2** 用拟牛顿法求解问题  $\min f(x) = x^5 + x^3 - 7x$  的最优解。

**解** 由于  $f'(x) = 5x^4 + 3x^2 - 7$ ,  $f'(0) = -7$ ,  $f'(1) = 1$ , 可知在  $[0,1]$  内有一驻点  $x^*$ , 可取  $[a,b] = [0,1]$ , 则

$$x_1 = a - \frac{(b-a)f'(a)}{f'(b) - f'(a)} = 0 - \frac{(1-0)(-7)}{1 - (-7)} = \frac{7}{8} \approx 0.8750, f'(x_1) \approx -1.7722$$

取  $[0.8750, 1]$ , 则

$$x_2 = x_1 - \frac{(1-x_1)f'(x_1)}{f'(b) - f'(x_1)} = 0.8750 - \frac{(1-0.8750)(-1.7722)}{1 - (-1.7722)} \approx 0.9549, f'(x_2) \approx -0.1073$$

取  $[0.9549, 1]$ , 则

$$x_3 = x_2 - \frac{(1-x_2)f'(x_2)}{f'(b) - f'(x_2)} = \cdots = 0.9593, f'(x_3) \approx -0.0048$$

由于  $f'(x_3) \approx 0$ , 再由  $f''(x_3) > 0$  可知,  $x^* \approx x_3 = 0.9593$  为最优解。

### 11.2.3 黄金分割法

黄金分割法（也称 0.618 法）的基本思想是：在区间  $[a, b]$  内选取两个对称点  $\lambda, u$  且  $\lambda < u$ ，通过比较  $f(\lambda), f(u)$  的大小来决定删除左半区间  $[a, \lambda]$ ，还是删除右半区间  $[u, b]$ 。删除后的新区间长度是原区间长度的 0.618。新区间包含原区间中的两个对称点中的一点，只要再选一个对称点，并利用这两个新对称点处的函数值继续比较，就可以重复这个过程。最后确定出极小点  $x^*$ 。黄金分割法前提是，目标函数  $f(x)$  为单峰函数。

记  $a_0 = a, b_0 = b$ ，区间  $[a_0, b_0]$  经  $k$  次缩短后变为  $[a_k, b_k]$ ，现要求两个试探点  $\lambda_k, u_k$  满足条件

$$b_k - \lambda_k = u_k - a_k \quad (11.2.4)$$

$$b_{k+1} - a_{k+1} = \tau(b_k - a_k) \quad (11.2.5)$$

由式 (11.2.4) 和式 (11.2.5) 得

$$\lambda_k = a_k + (1-\tau)(b_k - a_k) \quad (11.2.6)$$

$$u_k = a_k + \tau(b_k - a_k) \quad (11.2.7)$$

计算  $f(\lambda_k), f(u_k)$ ：若  $f(\lambda_k) \leq f(u_k)$ ，则选取  $[a_k, u_k]$ ，删掉  $[u_k, b_k]$ ，新区间为  $[a_{k+1}, b_{k+1}] = [a_k, u_k]$ 。

为进一步缩短区间，需取试探点  $\lambda_{k+1}, u_{k+1}$ ，由式 (11.2.7) 计算得

$$\begin{aligned} u_{k+1} &= a_{k+1} + \tau(b_{k+1} - a_{k+1}) = a_k + \tau(u_k - a_k) \\ &= a_k + \tau(a_k + \tau(b_k - a_k) - a_k) = a_k + \tau^2(b_k - a_k) \end{aligned}$$

与式 (11.2.6) 对比，若令  $\tau^2 = 1 - \tau$ ，则  $u_{k+1} = a_k + (1-\tau)(b_k - a_k) = \lambda_k$ 。这样，新的试探点  $u_{k+1}$  不需要重新计算，只要取  $\lambda_k$  就行，从而每次迭代（第一次迭代除外）时，只需选取一个试探点即可。

类似地，对于  $f(\lambda_k) > f(u_k)$  的情形，新的试探点  $\lambda_{k+1} = u_k$  也不需要重新计算。此时保留  $[\lambda_k, b_k]$ ，即  $[a_{k+1}, b_{k+1}] = [\lambda_k, b_k]$ 。

由  $\tau^2 = 1 - \tau$ ，求解得  $\tau = \frac{\sqrt{5}-1}{2} = 0.618$ 。因此，黄金分割法也称 0.618 法。

黄金分割法的算法描述如下。

设有问题  $\min f(x)$ ， $f(x)$  在区间  $[a_1, b_1]$  内连续，下单峰，令  $\tau = 0.618$ ， $\varepsilon = 0$ ，分步操作如下。

① 令  $k=1$ ，计算

$$\lambda_k = a_k + (1-\tau)(b_k - a_k) \text{ 与 } f(\lambda_k)$$

$$u_k = a_k + \tau(b_k - a_k) \text{ 与 } f(u_k)$$

② 若  $b_k - a_k < \varepsilon$ ，则计算结束，最优解  $\lambda^* \in [a_k, b_k]$ ，可取  $x^* = (b_k + a_k)/2$ 。否则，对于  $f(\lambda_k) > f(u_k)$ ，转③，反之则转④。

③ 令  $a_{k+1} = \lambda_k$ ， $b_{k+1} = b_k$ ，再令  $\lambda_{k+1} = u_k, u_{k+1} = a_{k+1} + \tau(b_{k+1} - a_{k+1})$ ，计算  $f(u_{k+1})$ ，转⑤。

④ 令  $a_{k+1} = a_k$ ， $b_{k+1} = u_k$ ，再令  $u_{k+1} = \lambda_k$ ， $\lambda_{k+1} = a_{k+1} + (1-\tau)(b_{k+1} - a_{k+1})$ ，计算  $f(\lambda_{k+1})$ ，转⑤。

⑤ 令  $k = k+1$ ，返回②。

**例 11.2.3** 用 0.618 法求解问题  $\min f(x) = 2x^2 - x - 1$  的最优解。初始区间  $[a_1, b_1] = [-1, 1]$ ，精度  $\varepsilon = 0.15$ 。

解 计算结果见表 11.2.1

表 11.2.1 例 11.2.3 的计算结果

$k$	$a_k$	$b_k$	$\lambda_k$	$\mu_k$	$f(\lambda_k)$	$f(\mu_k)$
1	-1	1	-0.236	0.236	-0.653	-1.125
2	-0.236	1	0.236	0.582	-1.125	-0.970
3	-0.236	0.582	0.056	0.236	-1.050	-1.125
4	0.056	0.582	0.236	0.348	-1.125	-1.106
5	0.056	0.348	0.168	0.236	-1.112	-1.125
6	0.168	0.348	0.236	0.279	-1.125	-1.123
7	0.168	0.279				

经过 6 次迭代, 得  $b_7 - a_7 = 0.111 < 0.15$ , 满足精度要求, 最优近似解  $x_7 \in [0.618, 0.279] = \frac{1}{2}(0.618 + 0.279) \approx 0.23$ 。

实际上, 问题的最优解  $x^* = 0.25$ , 可将  $x_7$  作为  $x^*$  的近似解。

黄金分割法的特点是: 要求  $f(x)$  为单峰即可, 不要求连续可导。迭代时只求函数值, 不要求导数值。但其收敛速度比牛顿法慢。

## 11.3 多元函数优化计算

### 11.3.1 多元函数有最优解的条件

设有问题

$$\min f(\mathbf{x}), \mathbf{x} = (x_1, x_2, \dots, x_n)^T \in R^n \quad (11.3.1)$$

问题 (11.3.1) 有解的必要条件是: 若  $\mathbf{x}^*$  是问题 (11.3.1) 的局部最优解, 则  $\mathbf{x}^*$  必使  $f(\mathbf{x})$  在  $\mathbf{x}^*$  点的梯度向量

$$\nabla f(\mathbf{x}^*) = (\partial f_{x_1}, \partial f_{x_2}, \dots, \partial f_{x_n})^T = 0 \quad (11.3.2)$$

问题 (11.3.1) 有解的充分条件是: 若  $\mathbf{x}^*$  使  $\nabla f(\mathbf{x}^*) = 0$ , 且  $\nabla^2 f(\mathbf{x}^*) > 0$ , 则  $\mathbf{x}^*$  是问题 (11.3.1) 的局部最优解。其中,

$$\nabla^2 f(\mathbf{x}^*) = \begin{pmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ f_{n1} & f_{n2} & \cdots & f_{nn} \end{pmatrix} = \mathbf{H}(\mathbf{x}^*) \quad (11.3.3)$$

是  $f(\mathbf{x})$  在  $\mathbf{x}^*$  点处的 Hesse 矩阵。用  $\mathbf{H}(\mathbf{x}^*) = \nabla^2 f(\mathbf{x}^*) > 0$  表示矩阵  $\mathbf{H}(\mathbf{x})$  在  $\mathbf{x}^*$  点处正定。对于  $n=2$  时, 二元函数  $z = f(x, y)$  对应一张 (单值) 曲面  $S$ 。必要条件  $\nabla f(\mathbf{x}^*) = 0$ , 对应于在点  $\mathbf{x}^*$  处, 曲面  $S$  的切平面是水平的。充分条件对应于在点  $\mathbf{x}^*$  处, 不仅切平面水平, 同时曲面  $S$  还是向下凸的。

例 11.3.1 讨论  $z = (x-a)^2 + (y-b)^2$  的最优解。

解 因为  $\nabla f(\mathbf{x}, \mathbf{y}) = (2(x-a), 2(y-b))^T$ ,  $\nabla f(\mathbf{a}, \mathbf{b}) = (0, 0)^T$

$$H(\mathbf{x}, \mathbf{y}) = \nabla^2 f(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} > 0, \text{ 正定}$$

所以  $z = (\mathbf{x} - \mathbf{a})^2 + (\mathbf{y} - \mathbf{b})^2$  的最优解为  $(0, 0)$ ，也是严格全局最优解。

**例 11.3.2** 讨论  $z = (\mathbf{x} - \mathbf{y})^2$  的最优解。

**解** 因为  $\nabla f(\mathbf{x}, \mathbf{y}) = (2x - 2y, -2x + 2y)^T$ ，在直线  $\mathbf{y} = \mathbf{x}$  上， $\nabla f(\mathbf{x}, \mathbf{y}) = (0, 0)^T$ ，即曲面  $S$  在直线  $\mathbf{y} = \mathbf{x}$  上的切平面是水平的。

$$H(\mathbf{x}, \mathbf{y}) = \nabla^2 f(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} 2 & -2 \\ -2 & 2 \end{pmatrix} \geq 0, \text{ 半正定}$$

所以曲面  $S$  在直线  $\mathbf{y} = \mathbf{x}$  上取得局部最优解。

**例 11.3.3** 讨论  $z = \mathbf{x}^3 + \mathbf{y}^3 - 3\mathbf{x} - 12\mathbf{y}$  的极值。

**解** 由于  $\nabla f(\mathbf{x}, \mathbf{y}) = (3x^2 - 3, 3y^2 - 12)^T$

令  $\nabla f(\mathbf{x}, \mathbf{y}) = 0$ ，求解方程组  $\begin{cases} 3(x^2 - 1) = 0 \\ 3(y^2 - 4) = 0 \end{cases}$ ，得  $A(1, 2), B(1, -2), C(-1, 2), D(-1, -2)$  四个点。

$$H(\mathbf{x}, \mathbf{y}) = \nabla^2 f(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} 6x & 0 \\ 0 & 6y \end{pmatrix}$$

$$\text{一阶主子式 } f_{xx} = 6x, \text{ 二阶主子式 } \begin{vmatrix} 6x & 0 \\ 0 & 6y \end{vmatrix} = 36xy$$

当  $6x > 0$ ，且  $36xy > 0$  时，得  $x > 0, y > 0$ ，为第一象限，可知在第一象限里  $H(\mathbf{x}, \mathbf{y})$  正定，曲面  $S$  为下凸，又因为  $\nabla f(\mathbf{A}) = \nabla f(1, 2) = 0$  且  $A(1, 2)$  点在第一象限内，从而可知  $A(1, 2)$  为曲面  $S$  的一个严格局部最优解。

## 11.3.2 多元函数数值求解的原则

多元函数数值求解的基本思想是：设问题 (11.3.1) 的某个局部最优解是  $\mathbf{x}^*$ ，先给  $\mathbf{x}^*$  的一个初始估计  $\mathbf{x}_0$ ，称为初始点，然后从  $\mathbf{x}_0$  出发，按照某种规律计算出一系列的点  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k, \dots$  或  $\{\mathbf{x}_k\} (k=1, 2, \dots)$ ，并希望当  $k \rightarrow \infty$  时， $\{\mathbf{x}_k\} \rightarrow \mathbf{x}^*$ 。

现在关键的问题是，如何构造点列  $\{\mathbf{x}_k\}$ 。由于  $\mathbf{x}_{k+1} - \mathbf{x}_k$  是一个向量，这个向量应该由它的单位方向矢量  $\bar{s}_k$  与它的长度  $t_k \geq 0$  来表示，即  $\mathbf{x}_{k+1} - \mathbf{x}_k = t_k \cdot \bar{s}_k$  或  $\mathbf{x}_{k+1} = \mathbf{x}_k + t_k \cdot \bar{s}_k$ 。由此可知， $\{\mathbf{x}_k\}$  的产生完全由它的方向矢量  $\bar{s}_k$  和长度  $t_k \geq 0$  这两个要素确定，尤其是  $\bar{s}_k$  起着更加关键的作用。一般， $\bar{s}_k$  与  $t_k$  选取的原则如下。

① 下降点列（下降算法）。对应于点列  $\{\mathbf{x}_k\}$  的函数值  $\{f(\mathbf{x}_k)\}$  必须是逐渐减小的或者至少不增加，即有  $f(\mathbf{x}_0) \geq f(\mathbf{x}_1) \geq \dots \geq f(\mathbf{x}_k) \geq f(\mathbf{x}_{k+1}) \geq \dots$ 。具有这种性质的点列称为下降点列，产生这种点列的方法（算法）称为下降算法。

② 收敛点列（收敛算法）。对于下降点列  $\{\mathbf{x}_k\}$  还要求具有性质：当  $k \rightarrow \infty$  时  $\{\mathbf{x}_k\} \rightarrow \mathbf{x}^*$ ，即  $\{\mathbf{x}_k\}$  收敛于  $\mathbf{x}^*$ ，具有这种性质的点列称为收敛点列，相应的方法称为收敛算法。

构造下降、收敛算法的步骤如下。

① 选取初始点  $\mathbf{x}_0$ ，指定计算精度  $\varepsilon > 0$ 。

② 选定某种收敛算法，即确定迭代方向矢量  $\bar{s}_k$  与确定的搜索步长  $t_k > 0$ （通常用一维搜索

$\min_{t \geq 0} f(\mathbf{x}_k + t\bar{s}_k)$  来确定步长  $t_k > 0$  )。

③ 由  $\bar{s}_k$  与  $t_k$  产生下一个新点  $\mathbf{x}_{k+1} = \mathbf{x}_k + t_k \bar{s}_k$ 。

④ 检查  $\|\nabla f(\mathbf{x}_{k+1})\| < \varepsilon$  是否成立, 若成立, 则停止计算, 令  $\mathbf{x}^* = \mathbf{x}_{k+1}$ ; 否则令  $k = k+1$ , 转③。

这里有两个核心问题: 首先是算法的收敛速度问题, 在函数优化算法中, 收敛速度是用收敛的阶来定量描述的。

**定义 11.3.1 (收敛阶)** 设  $\mathbf{x}^*$  为问题 (11.3.1) 的最优解, 由某算法产生的序列  $\{\mathbf{x}_k\}$  收敛于  $\mathbf{x}^*$ , 即有  $\lim_{k \rightarrow \infty} \mathbf{x}_k = \mathbf{x}^*$ 。如果存在同  $k$  无关的常数  $\alpha \geq 0$ ,  $\beta > 0$ , 使得

$$\lim_{k \rightarrow \infty} \frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|^\alpha} = \beta \quad (11.3.4)$$

成立, 则称序列  $\{\mathbf{x}_k\}$  是  $\alpha$  阶收敛的, 或说该算法是  $\alpha$  阶收敛的。

特别地, 当  $\alpha = 1$  时, 称该算法是一阶 (或线性) 收敛的; 当  $\alpha = 2$  时, 称该算法是二阶 (或平方) 收敛的; 当  $1 < \alpha < 2$  时, 称该算法是超线性收敛的。

一般来说, 一阶收敛速度较慢, 二阶收敛速度较快, 但算法实现时困难较大, 而超线性收敛速度适中, 在具体应用时难度适中, 应用较广泛。

其次, 是算法的结束准则问题, 设  $\varepsilon_1 > 0$ ,  $\varepsilon_2 > 0$  为预先给定的正小数, 常用的结束准则如下。

① 若  $|\nabla f(\mathbf{x}_k)| < \varepsilon_1$  成立, 则迭代结束, 令  $\mathbf{x}^* \approx \mathbf{x}_k$ 。

② 若  $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| < \varepsilon_1$  成立, 则迭代结束, 令  $\mathbf{x}^* \approx \mathbf{x}_k$ ; 或者, 若  $\frac{\|\mathbf{x}_{k+1} - \mathbf{x}_k\|}{\|\mathbf{x}_k\|} < \varepsilon_2$  成立, 则迭代结束, 令  $\mathbf{x}^* \approx \mathbf{x}_k$ 。

③ 若  $|f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)| < \varepsilon_1$  成立, 则迭代结束, 令  $\mathbf{x}^* \approx \mathbf{x}_k$ ; 或者, 若  $\frac{|f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)|}{|f(\mathbf{x}_k)|} < \varepsilon_2$  成立, 则迭代结束, 令  $\mathbf{x}^* \approx \mathbf{x}_k$ 。

④ 若  $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| < \varepsilon_1$  与  $|f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)| < \varepsilon_2$  同时成立, 则迭代结束, 令  $\mathbf{x}^* \approx \mathbf{x}_k$ 。

### 11.3.3 梯度法

先以二元函数为例, 讨论梯度法的基本思想, 所得结论不难推广到多元函数的情况。设有二元函数  $f(\mathbf{x}, \mathbf{y})$ , 其偏导数为

$$\frac{\partial f}{\partial x} = \lim_{\Delta x \rightarrow 0} \frac{f(\mathbf{x}_0 + \Delta \mathbf{x}, \mathbf{y}_0) - f(\mathbf{x}_0, \mathbf{y}_0)}{\Delta x}, \quad \frac{\partial f}{\partial y} = \lim_{\Delta y \rightarrow 0} \frac{f(\mathbf{x}_0, \mathbf{y}_0 + \Delta \mathbf{y}) - f(\mathbf{x}_0, \mathbf{y}_0)}{\Delta y}$$

方向导数为  $\frac{\partial f}{\partial l} = \lim_{\rho \rightarrow 0} \frac{f(\mathbf{x} + \Delta \mathbf{x}, \mathbf{y} + \Delta \mathbf{y}) - f(\mathbf{x}, \mathbf{y})}{\rho}, \quad \rho = \sqrt{(\Delta \mathbf{x})^2 + (\Delta \mathbf{y})^2}$

且  $\frac{\partial f}{\partial l} = \frac{\partial f}{\partial x} \cos \varphi + \frac{\partial f}{\partial y} \sin \varphi$ , 其中  $\varphi$  为  $x$  轴到方向  $l$  的转角。

梯度为  $\text{grad} f(\mathbf{x}, \mathbf{y}) = \frac{\partial f}{\partial x} \vec{i} + \frac{\partial f}{\partial y} \vec{j} = (f_x, f_y)^T$ , 设  $\mathbf{e}$  是  $l$  方向上的单位向量, 即  $\mathbf{e} = \cos \varphi \cdot \vec{i} + \sin \varphi \cdot \vec{j}$ ,

则

$$\begin{aligned}\frac{\partial f}{\partial l} &= \frac{\partial f}{\partial x} \cos \varphi + \frac{\partial f}{\partial y} \sin \varphi = \left( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) \cdot (\cos \varphi, \sin \varphi) = \text{grad } f(\mathbf{x}, \mathbf{y}) \cdot \mathbf{e} \\ &= |\text{grad } f(\mathbf{x}, \mathbf{y})| \cos(\text{grad } f(\mathbf{x}, \mathbf{y}), \mathbf{e})\end{aligned}$$

即方向导数  $\frac{\partial f}{\partial l}$  是梯度  $\text{grad } f(\mathbf{x}, \mathbf{y})$  在  $l$  方向上的投影。当方向  $l$  与梯度方向一致时,

$\cos(\text{grad } f(\mathbf{x}, \mathbf{y}), \mathbf{e}) = 1$ , 从而  $\frac{\partial f}{\partial l}$  有最大值, 即  $f(\mathbf{x}, \mathbf{y})$  增长最快。

这说明梯度矢量的一个重要性质是, 多元函数在某点处的梯度正方向时, 函数值将以最快速度增大, 而沿梯度负方向时, 函数值必以最快速度下降。由此可以得出梯度法的基本思想如下。

在  $\mathbf{x}^0 = (x_1^0, x_2^0)^T$  点选取负梯度方向  $\bar{\mathbf{s}} = -\nabla f(x_1^0, x_2^0)$  作为迭代方向, 则目标函数值下降最快, 即取

$$\mathbf{x} = \mathbf{x}^0 + t \cdot \bar{\mathbf{s}}_0 = \mathbf{x}^0 - t \nabla f(x_1^0, x_2^0)$$

为确定  $t$  值, 沿射线方向  $\mathbf{x}^0 - t \cdot \bar{\mathbf{s}}_0 (t \geq 0)$  进行一维搜索, 即求  $\min_{t \geq 0} f(\mathbf{x}^0 - t \cdot \bar{\mathbf{s}}_0)$ 。求得的  $t$  记为  $t_0$ , 从而得新点  $\mathbf{x}^1 = \mathbf{x}^0 - t_0 \cdot \bar{\mathbf{s}}_0$ 。同理可得一系列最快速度下降点列

$$\mathbf{x}^{k+1} = \mathbf{x}^k - t_k \nabla f(\mathbf{x}^k), \quad k = 0, 1, \dots \quad (11.3.5)$$

可以证明: 在一定条件下, 由式 (11.3.5) 得到的最快速度下降点列  $\{\mathbf{x}^k\}$ , 当  $k \rightarrow +\infty$  时, 收敛于函数  $f(x, y)$  的某个局部极小点  $\mathbf{x}^*$ , 且为一阶收敛。

梯度法的算法步骤如下。

- ① 给出初始点  $\mathbf{x}^0$  和计算精度  $\varepsilon > 0$ , 并令  $k = 0$ 。
- ② 计算  $\nabla f(\mathbf{x}^k)$ 。
- ③ 若  $\|\nabla f(\mathbf{x}^k)\| < \varepsilon$ , 则迭代结束, 取  $\mathbf{x}^* = \mathbf{x}^k$ , 否则转④。
- ④ 进行一维搜索, 由  $\min_{t \geq 0} f(\mathbf{x}) = \min_{t \geq 0} f(\mathbf{x}^k - t \nabla f(\mathbf{x}^k))$  求得  $t$ , 记为  $t_k$ 。
- ⑤ 令  $\mathbf{x}^{k+1} = \mathbf{x}^k - t_k \nabla f(\mathbf{x}^k)$ ,  $k = k + 1$ , 返回②。

**例 11.3.4** 用梯度法求解问题  $\min f(\mathbf{x}) = 0.8x_1^2 + x_2^2$ , 取  $\mathbf{x}^0 = (1, 1)$ ,  $\varepsilon = 0.05$ 。

**解** 第一轮迭代如下。

- ① 初值  $\mathbf{x}^0 = (1, 1)^T$ ,  $\varepsilon = 0.05$ , 令  $k = 0$ 。
- ② 计算  $\nabla f(\mathbf{x}) = (1.6x_1, 2x_2)^T$ ,  $\nabla f(\mathbf{x}^0) = (1.6, 2)^T$ 。
- ③ 判断  $\|\nabla f(\mathbf{x}^0)\| = \sqrt{1.6^2 + 2^2} = 2.56 > \varepsilon$ 。
- ④ 进行一维搜索。

$$\begin{aligned}\mathbf{x} &= \mathbf{x}^0 - t \nabla f(\mathbf{x}^0) = \begin{pmatrix} 1 \\ 1 \end{pmatrix} - t \begin{pmatrix} 1.6 \\ 2.0 \end{pmatrix} = \begin{pmatrix} 1 - 1.6t \\ 1 - 2t \end{pmatrix} \\ \min f(\mathbf{x}) &= \min_{t \geq 0} f(1 - 1.6t, 1 - 2t) = \min_{t \geq 0} (0.8(1 - 1.6t)^2 + (1 - 2t)^2)\end{aligned}$$

求得  $t_0 = 0.54$ 。

- ⑤ 由此,  $\mathbf{x}^1 = \begin{pmatrix} 1 - 1.6 \times 0.54 \\ 1 - 2 \times 0.54 \end{pmatrix} = \begin{pmatrix} 0.136 \\ -0.081 \end{pmatrix}$ , 返回②



第二轮迭代如下。

$$\textcircled{1} \text{ 计算 } \nabla f(\mathbf{x}^1) = \begin{pmatrix} 1.6 \times 0.136 \\ 2 \times (-0.081) \end{pmatrix} = \begin{pmatrix} 0.217 \\ -0.160 \end{pmatrix}$$

$$\textcircled{2} \text{ 判断 } \|\nabla f(\mathbf{x}^1)\| = \sqrt{0.217^2 + (-0.16)^2} = 0.26 > \varepsilon$$

③ 进行一维搜索

$$\begin{aligned} \mathbf{x} &= \mathbf{x}^1 - t \nabla f(\mathbf{x}^1) = \begin{pmatrix} 0.136 \\ -0.081 \end{pmatrix} - \begin{pmatrix} 0.217t \\ -0.160t \end{pmatrix} = \begin{pmatrix} 0.136 - 0.217t \\ -0.081 + 0.160t \end{pmatrix} \\ \min_{t \geq 0} f(\mathbf{x}) &= \min_{t \geq 0} f(0.136 - 0.217t, -0.081 + 0.16t) \\ &= \min_{t \geq 0} [0.8(0.136 - 0.217t)^2 + (-0.081 + 0.16t)^2] \end{aligned}$$

求得  $t_1 = 0.58$ 。

$$\textcircled{4} \text{ 由此, } \mathbf{x}^2 = \begin{pmatrix} 0.136 - 0.217 \times 0.58 \\ -0.081 + 0.16 \times 0.58 \end{pmatrix} = \begin{pmatrix} 0.0124 \\ 0.0128 \end{pmatrix}$$

第三轮迭代如下。

$$\textcircled{1} \text{ 计算 } \nabla f(\mathbf{x}^2) = \begin{pmatrix} 1.6 \times 0.0124 \\ 2 \times 0.0128 \end{pmatrix} = \begin{pmatrix} 0.0198 \\ 0.0256 \end{pmatrix}$$

$$\textcircled{2} \text{ 判断 } \|\nabla f(\mathbf{x}^2)\| = \sqrt{0.0198^2 + 0.0256^2} = 0.03236 < \varepsilon, \text{ 结束。}$$

由于精确最优解是  $\mathbf{x}^* = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ , 因此用  $\mathbf{x}^* \approx \mathbf{x}^2 = \begin{pmatrix} 0.0124 \\ 0.0128 \end{pmatrix}$  是合理的。

梯度法的优缺点如下。

优点: ①方法简单, 每一轮迭代的计算量较小, 存储量也较小。②对初始点的要求很少, 从一般的初始点出发, 都能收敛到某个局部极小点。

缺点: ①梯度法的收敛速度与变量的尺度关系很大, 对有些例子, 在极小点附近会产生显著的锯齿现象, 收敛十分缓慢。②梯度法对于小的扰动是不稳定的, 而小扰动在计算中常常不可避免。③梯度法的最快速度下降仅是一种局部性质, 即从局部看, 目标函数值下降得最快, 但从总体看, 它可能走了许多弯路, 主要是因为每相邻的两个方向都是正交的。

### 11.3.4 牛顿法

对于问题 (11.3.1)

$$\min f(\mathbf{x}), \mathbf{x} = (x_1, x_2, \dots, x_n)^T \in R^n$$

设多元函数  $f(\mathbf{x})$  连续并至少二阶可微, 对  $f(\mathbf{x})$  在  $\mathbf{x}_0$  点进行二阶泰勒展开, 得

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \frac{1}{1!} \nabla f(\mathbf{x}_0)^T (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2!} (\mathbf{x} - \mathbf{x}_0)^T \nabla^2 f(\mathbf{x}_0) (\mathbf{x} - \mathbf{x}_0) + O(\|\mathbf{x} - \mathbf{x}_0\|^2)$$

$$\text{令 } g(\mathbf{x}) = f(\mathbf{x}_0) + \frac{1}{1!} \nabla f(\mathbf{x}_0)^T (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2!} (\mathbf{x} - \mathbf{x}_0)^T \nabla^2 f(\mathbf{x}_0) (\mathbf{x} - \mathbf{x}_0) \approx f(\mathbf{x})$$

$$\text{求 } \nabla g(\mathbf{x}) = \nabla f(\mathbf{x}_0) + \nabla^2 f(\mathbf{x}_0) (\mathbf{x} - \mathbf{x}_0)$$

$$\text{令 } \nabla g(\mathbf{x}) = 0 \text{ 解出 } \mathbf{x} \text{ 有 } \mathbf{x} = \mathbf{x}_0 - [\nabla^2 f(\mathbf{x}_0)]^{-1} \nabla f(\mathbf{x}_0)$$

$$\text{或 } \mathbf{x}_{k+1} = \mathbf{x}_k - [\nabla^2 f(\mathbf{x}_k)]^{-1} \nabla f(\mathbf{x}_k) \quad (11.3.6)$$

式 (11.3.6) 称为多元牛顿 (Newton) 迭代公式。式中,  $[\nabla^2 f(\mathbf{x}_k)]^{-1}$  为 Hesse 矩阵的逆矩阵, 向量  $\bar{s} = -[\nabla^2 f(\mathbf{x}_k)]\nabla f(\mathbf{x}_k)$  称为牛顿方向。

牛顿法的算法步骤如下。

- ① 给出初始点  $\mathbf{x}^0$  和计算精度  $\varepsilon > 0$ , 并令  $k = 0$ 。
- ② 计算  $\nabla f(\mathbf{x}^k)$ , 如果  $\|\nabla f(\mathbf{x}^k)\| < \varepsilon$ , 则迭代结束, 取  $\mathbf{x}^* = \mathbf{x}^k$ , 否则转③。
- ③ 计算  $[\nabla^2 f(\mathbf{x}_k)]^{-1}$  与  $[\nabla^2 f(\mathbf{x}_k)]^{-1}\nabla f(\mathbf{x}_k)$ 。
- ④ 令  $\mathbf{x}^{k+1} = \mathbf{x}^k - [\nabla^2 f(\mathbf{x}_k)]^{-1}\nabla f(\mathbf{x}_k)$ ,  $k = k + 1$ , 返回②。

**例 11.3.5** 用牛顿法求解问题  $\min f(\mathbf{x}) = \min f(x_1, x_2) = x_1^2 + 25x_2^2$  的最优解。

**解** ①取初值  $\mathbf{x}^0 = (1, 1)^T$ ,  $\varepsilon = 0.05$ , 令  $k = 0$

②计算  $\nabla f(\mathbf{x}) = (2x_1, 50x_2)^T$ ,  $\nabla f(\mathbf{x}^0) = (2, 50)^T$ , 判断  $\|\nabla f(\mathbf{x}^0)\| = \sqrt{2^2 + 50^2} > \varepsilon$  转③。

③计算  $\nabla^2 f(\mathbf{x}^0) = \begin{pmatrix} 2 & 0 \\ 0 & 50 \end{pmatrix}$ ,  $[\nabla^2 f(\mathbf{x}^0)]^{-1} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{50} \end{pmatrix}$  和

$$[\nabla^2 f(\mathbf{x}^0)]^{-1}\nabla f(\mathbf{x}^0) = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{50} \end{pmatrix} \begin{pmatrix} 2 \\ 50 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

④令  $\mathbf{x}^1 = \mathbf{x}^0 - [\nabla^2 f(\mathbf{x}^0)]^{-1}\nabla f(\mathbf{x}^0) = \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ , 从而求得  $\mathbf{x}^* = \mathbf{x}^1 = (0, 0)^T$ 。

可以证明, 牛顿法对于所有二次函数都是精确成立的。

**例 11.3.6** 用牛顿法求解问题  $\min f(\mathbf{x}) = \min f(x_1, x_2) = x_1^3 + x_2^3 - 3(x_1 + x_2)$  的最优解, 取  $\mathbf{x}^0 = (6, 4)^T$ ,  $\varepsilon = 0.05$ 。

**解** ①初值  $\mathbf{x}^0 = (6, 4)^T$ ,  $\varepsilon = 0.05$ , 令  $k = 0$ 。

②计算  $\nabla f(\mathbf{x}) = 3 \begin{pmatrix} x_1^2 - 1 \\ x_2^2 - 1 \end{pmatrix}$ ,  $\nabla f(\mathbf{x}^0) = 3 \begin{pmatrix} 35 \\ 15 \end{pmatrix}$ , 判断  $\|\nabla f(\mathbf{x}^0)\| = 3\sqrt{35^2 + 15^2} > \varepsilon$ , 转③。

③计算  $\nabla^2 f(\mathbf{x}) = \begin{pmatrix} 6x_1 & 0 \\ 0 & 6x_2 \end{pmatrix}$ ,  $[\nabla^2 f(\mathbf{x})] = \begin{pmatrix} \frac{1}{6x_1} & 0 \\ 0 & \frac{1}{6x_2} \end{pmatrix}$ ,  $[\nabla^2 f(\mathbf{x}^0)]^{-1} = \begin{pmatrix} \frac{1}{36} & 0 \\ 0 & \frac{1}{24} \end{pmatrix}$  和

$$[\nabla^2 f(\mathbf{x}^0)]^{-1}\nabla f(\mathbf{x}^0) = 3 \begin{pmatrix} \frac{1}{36} & 0 \\ 0 & \frac{1}{24} \end{pmatrix} \begin{pmatrix} 35 \\ 15 \end{pmatrix} = \frac{1}{25} \begin{pmatrix} 14 \\ 9 \end{pmatrix}$$

④令  $\mathbf{x}^1 = \mathbf{x}^0 - [\nabla^2 f(\mathbf{x}^0)]^{-1}\nabla f(\mathbf{x}^0) = \begin{pmatrix} 6 \\ 4 \end{pmatrix} - \frac{1}{25} \begin{pmatrix} 14 \\ 9 \end{pmatrix} = \begin{pmatrix} 3.0833 \\ 2.1250 \end{pmatrix}$ 。

重复以上步骤可得

$$\mathbf{x}^2 = \begin{pmatrix} 1.7038 \\ 1.2978 \end{pmatrix}, \quad \mathbf{x}^3 = \begin{pmatrix} 1.1454 \\ 1.0342 \end{pmatrix}, \quad \mathbf{x}^4 = \begin{pmatrix} 1.0092 \\ 1.0006 \end{pmatrix}, \quad \mathbf{x}^5 = \begin{pmatrix} 1.00004190 \\ 1.00000018 \end{pmatrix} \rightarrow \mathbf{x}^* = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

牛顿法的优缺点如下。

优点：收敛速度快，为二阶收敛。

缺点：要求  $f(\mathbf{x})$  二阶可导；在迭代中要多次计算  $[\nabla^2 f(\mathbf{x})]^{-1}$ ，计算非常困难；对初始点  $\mathbf{x}^0$  要求有限制， $\mathbf{x}^0$  必须在某个凸区间内。

### 11.3.5 共轭方向法

共轭方向法兼顾梯度法和牛顿法二者的优点，又避开它们的缺点。它收敛速度快，且不需要求解  $[\nabla^2 f(\mathbf{x})]^{-1}$ 。

**定义 11.3.2** 设  $A_{n \times n}$  为  $n$  阶对称满秩方阵， $\mathbf{x}_1, \mathbf{x}_2$  为  $n$  维向量，如果  $\mathbf{x}_1^T A \mathbf{x}_2 = 0$ ，则称  $\mathbf{x}_1, \mathbf{x}_2$  关于  $A$  共轭。若向量组  $\{\mathbf{x}_k\} (k=1, 2, \dots)$  满足  $\mathbf{x}_i^T A \mathbf{x}_j = 0 (i \neq j; i, j=1, 2, \dots, m \leq n)$ ，则称向量组  $\{\mathbf{x}_k\} (k=1, 2, \dots)$  为  $A$  的共轭向量组。

特别地，当  $A = I$  为单位向量时， $\mathbf{x}_1^T \mathbf{x}_2 = 0$ ， $\mathbf{x}_1, \mathbf{x}_2$  正交，这说明共轭方向是正交方向的推广。可以证明，共轭方向有如下两个性质。

① 设  $A$  为  $n$  阶实对称满秩矩阵， $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n$  是关于  $A$  共轭的非零向量组，则此向量组必定是线性无关的。

② 设  $A$  为  $n$  元二次函数  $f(\mathbf{x})$  二次项的系数矩阵，且  $A$  满秩，则从任一初始点  $\mathbf{x}^0 = (x_1^0, x_2^0, \dots, x_n^0)$  出发，如果依次沿一组  $A$  的共轭方向  $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n$  进行一维最优搜索，则最多迭代  $n$  次（可少于  $n$  次），就可到达  $f(\mathbf{x})$  的逗留点（此性质也称为二次终止性质）。

由共轭方向性质②可知，对于  $n$  元二次函数  $f(\mathbf{x})$  的优化问题

$$\min f(\mathbf{x}) = c + b^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T A \mathbf{x}, \quad \mathbf{x} = (x_1, x_2, \dots, x_n)^T$$

只要找到一组（ $n$  个）关于  $A$  的共轭方向  $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n$ ，从任一初始点  $\mathbf{x}^0$  出发，依序沿这组方向进行一维搜索，就可以在  $n$  步之内求得此问题的最优解。对于一般的  $n$  元非线性连续可微函数  $f(\mathbf{x})$ ，虽然不存在一组（ $n$  个）关于  $A$  的共轭方向  $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n$  的定义，但总可在  $f(\mathbf{x})$  的某个局部极小点  $\mathbf{x}^*$  邻近的一点  $\mathbf{x}^k$  处将  $f(\mathbf{x})$  进行二阶泰勒展开，用展开后得到的多元二次函数  $g(\mathbf{x})$  去近似代替  $f(\mathbf{x})$ ，再利用  $g(\mathbf{x})$  的一组共轭方向进行搜索，将会收到较好的效果。但要达到计算的精度，一般无法在  $n$  步（一轮）之内完成，通常还要再做数轮。这就是共轭方向法的基本思想。下面讨论共轭方向的生成。

设有问题

$$\min f(\mathbf{x}) = c + b^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T H \mathbf{x}, \quad \mathbf{x} = (x_1, x_2, \dots, x_n)^T, \quad H > 0 \text{ 正定} \quad (11.3.7)$$

任取初始点  $\mathbf{x}^0$ ，第一个方向取  $\mathbf{s}^0 = -\nabla f(\mathbf{x}^0)$ ，此方法也称共轭梯度法。有  $\mathbf{x} = \mathbf{x}^0 + t\mathbf{s}^0 = \mathbf{x}^0 - t\nabla f(\mathbf{x}^0)$ ，进行一维搜索  $\min_{t \geq 0} f(\mathbf{x}^0 - t\nabla f(\mathbf{x}^0))$ ，求得的  $t$  值记为  $t_0$ ，从而可得  $\mathbf{x}^1 = \mathbf{x}^0 + t_0 \nabla f(\mathbf{x}^0)$ 。

第二个方向取为  $\mathbf{s}^1 = -\nabla f(\mathbf{x}^1) + \lambda_{10} \mathbf{s}^0$ ，即将  $\mathbf{s}^1$  取在  $-\nabla f(\mathbf{x}^1)$  与  $\mathbf{s}^0$  所张的平面上，式中  $\lambda_{10}$  待定，希望它能使得  $\mathbf{s}^1$  与  $\mathbf{s}^0$  关于  $H$  共轭，使得  $(\mathbf{s}^1)^T H \mathbf{s}^0 = 0$ ，于是有

$$(-\nabla f(\mathbf{x}^1) + \lambda_{10} \mathbf{s}^0)^T H \mathbf{s}^0 = 0$$

可得  $\lambda_{10} = \frac{\nabla f(\mathbf{x}^1) \mathbf{H} \mathbf{s}^0}{(\mathbf{s}^0)^T \mathbf{H} \mathbf{s}^0}$ 。

这样,  $\mathbf{s}^1$  便被确定, 且知  $\mathbf{s}^1$  与  $\mathbf{s}^0$  关于  $\mathbf{H}$  共轭。

再取  $\mathbf{x}^2 = \mathbf{x}^1 + t_1 \mathbf{s}^1$ , 式中的  $t_1$  由一维搜索  $\min_{t \geq 0} f(\mathbf{x}^1 + t \mathbf{s}^1)$  求得。

第三个方向取为

$$\mathbf{s}^2 = -\nabla f(\mathbf{x}^2) + \lambda_{20} \mathbf{s}^0 + \lambda_{21} \mathbf{s}^1$$

即将  $\mathbf{s}^2$  取在  $-\nabla f(\mathbf{x}^2)$ ,  $\mathbf{s}^0$ ,  $\mathbf{s}^1$  所张的三维空间中,  $\lambda_{20}, \lambda_{21}$  待定, 希望它们使  $\mathbf{s}^2$ ,  $\mathbf{s}^1$ ,  $\mathbf{s}^0$  关于  $\mathbf{H}$  共轭, 即使得  $(\mathbf{s}^2)^T \mathbf{H} \mathbf{s}^0 = 0$ ,  $(\mathbf{s}^2)^T \mathbf{H} \mathbf{s}^1 = 0$ 。于是有

$$(-\nabla f(\mathbf{x}^2) + \lambda_{20} \mathbf{s}^0 + \lambda_{21} \mathbf{s}^1)^T \mathbf{H} \mathbf{s}^0 = 0, \quad (-\nabla f(\mathbf{x}^2) + \lambda_{20} \mathbf{s}^0 + \lambda_{21} \mathbf{s}^1)^T \mathbf{H} \mathbf{s}^1 = 0$$

解之可得

$$\lambda_{20} = \frac{\nabla f(\mathbf{x}^2)^T \mathbf{H} \mathbf{s}^0}{(\mathbf{s}^0)^T \mathbf{H} \mathbf{s}^0}, \quad \lambda_{21} = \frac{\nabla f(\mathbf{x}^2)^T \mathbf{H} \mathbf{s}^1}{(\mathbf{s}^1)^T \mathbf{H} \mathbf{s}^1}$$

类似地, 可得

$$\begin{aligned} \mathbf{s}^i &= -\nabla f(\mathbf{x}^i) + \sum_{j=0}^{i-1} \lambda_{ij} \mathbf{s}^j \\ \lambda_{ij} &= \frac{\nabla f(\mathbf{x}^i)^T \mathbf{H} \mathbf{s}^j}{(\mathbf{s}^j)^T \mathbf{H} \mathbf{s}^j} \quad (0 \leq i \leq n-1; j=0, 1, 2, \dots, i-1) \end{aligned} \quad (11.3.8)$$

由此可得到一组 ( $n$  个) 关于  $\mathbf{H}$  的共轭方向  $\mathbf{s}^0, \mathbf{s}^1, \dots, \mathbf{s}^{n-1}$ , 但式 (11.3.8) 中用到了  $f(\mathbf{x})$  的 Hesse 矩阵  $\mathbf{H}$ , 计算量太大, 应尽量避免。

若令  $\mathbf{g}(\mathbf{x}) = \nabla f(\mathbf{x})$ ,  $\mathbf{g}_i = \mathbf{g}(\mathbf{x}^i) = \nabla f(\mathbf{x}^i)$ , 可以证明

$$\lambda_{10} = \frac{\|\mathbf{g}_1\|^2}{\|\mathbf{g}_0\|^2}, \lambda_{20} = 0, \lambda_{21} = \frac{\|\mathbf{g}_2\|^2}{\|\mathbf{g}_1\|^2}, \dots$$

一般地有,  $\lambda_{ij} = 0 \quad (0 \leq j \leq i-2)$

$$\lambda_{i,i-1} = \frac{\|\mathbf{g}_i\|^2}{\|\mathbf{g}_{i-1}\|^2} \quad (11.3.9)$$

共轭方向法的算法步骤如下。

- ① 给出初始点  $\mathbf{x}^0$  和计算精度  $\varepsilon > 0$ 。
- ② 计算  $\mathbf{g}_0 = \nabla f(\mathbf{x}^0)$ , 令  $\mathbf{s}^0 = -\mathbf{g}_0, k = 0$ 。
- ③ 进行一维搜索  $\min_{t \geq 0} f(\mathbf{x}^k + t \mathbf{s}^k)$ , 求得的  $t$  值, 记为  $t_k$ , 计算  $\mathbf{x}^{k+1} = \mathbf{x}^k + t_k \mathbf{s}^k$ ,  $\mathbf{g}_{k+1} = \nabla f(\mathbf{x}^{k+1})$
- ④ 如果  $\|\mathbf{g}_{k+1}\| < \varepsilon$ , 则迭代结束, 取  $\mathbf{x}^* = \mathbf{x}^{k+1}$ , 否则转⑤。
- ⑤ 如果  $k < n-1$ , 利用式 (11.3.9) 计算  $\lambda_{k+1,k}$ , 即

$$\lambda_{k+1,k} = \frac{\|\mathbf{g}_{k+1}\|^2}{\|\mathbf{g}_k\|^2}, \quad \mathbf{s}^{k+1} = -\mathbf{g}_{k+1} + \lambda_{k+1,k} \mathbf{s}^k$$

令  $k = k+1$ , 返回③。

如果  $k = n-1$ , 令  $\mathbf{x}^0 = \mathbf{x}^n$ , 返回②。

**例 11.3.7** 用共轭方向法求问题  $\min f(\mathbf{x}) = \min f(x_1, x_2) = x_1^2 + 25x_2^2$  的最优解, 取  $\mathbf{x}^0 = (1, 1)^T$ ,

$\varepsilon = 0.05$ 。

解 ①初值  $\mathbf{x}^0 = (1, 1)^T$ ,  $\varepsilon = 0.05$ 。

②计算  $\nabla f(\mathbf{x}) = (2x_1, 50x_2)^T$ ,  $\mathbf{g}_0 = \nabla f(\mathbf{x}_0) = (2, 50)^T$

令  $\mathbf{s}^0 = -\mathbf{g}_0 = (-2, -50)$ ,  $k = 0$

③进行一维搜索

$$\mathbf{x} = \mathbf{x}^0 + t\mathbf{s}^0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} + t \begin{pmatrix} -2 \\ -50 \end{pmatrix} = \begin{pmatrix} 1-2t \\ 1-50t \end{pmatrix}$$

由  $\min_{t \geq 0} f(\mathbf{x}) = \min_{t \geq 0} f(1-2t, 1-50t) = \min_{t \geq 0} [(1-2t)^2 + 25(1-50t)^2]$  求得  $t = 0.02003$ , 由此得

$$\mathbf{x}^1 = \begin{pmatrix} 1-2 \times 0.02003 \\ 1-50 \times 0.02003 \end{pmatrix} = \begin{pmatrix} 0.95994 \\ -0.0015 \end{pmatrix}$$

$$\mathbf{g}_1 = \nabla f(\mathbf{x}^1) = \begin{pmatrix} 1.91988 \\ -0.0750 \end{pmatrix}$$

④检查  $\|\nabla f(\mathbf{x}^1)\| = \sqrt{1.91988^2 + (-0.0750)^2} = 1.9213 > \varepsilon$ , 转⑤。

⑤计算  $\lambda_{10} = \frac{\|\mathbf{g}_1\|^2}{\|\mathbf{g}_0\|^2} = \frac{1.91988^2 + (-0.0750)^2}{2^2 + 50^2} = 0.00147$

$$\mathbf{s}^1 = -\mathbf{g}_1 + \lambda_{10}\mathbf{s}^0 = \begin{pmatrix} -1.91988 \\ 0.0750 \end{pmatrix} + 0.00147 \begin{pmatrix} -2 \\ -50 \end{pmatrix} = \begin{pmatrix} -1.9213 \\ 0.0015 \end{pmatrix}$$

返回③进行一维搜索,  $\mathbf{x} = \mathbf{x}^1 + t\mathbf{s}^1 = \begin{pmatrix} 0.95994 \\ -0.0015 \end{pmatrix} + t \begin{pmatrix} -1.9213 \\ 0.0015 \end{pmatrix} = \begin{pmatrix} 0.95994 - 1.9213t \\ -0.0015 + 0.0015t \end{pmatrix}$

由  $\min_{t \geq 0} f(\mathbf{x}) = \min_{t \geq 0} [(0.95994 - 1.9213t)^2 + 25 \times (-0.0015 + 0.0015t)^2]$  求得  $t_1 = 0.4996$ 。

$$\mathbf{x}^2 = \mathbf{x}^1 + t_1\mathbf{s}^1 = \begin{pmatrix} 0.95994 - 1.9213 \times 0.4996 \\ -0.0015 + 0.0015 \times 0.4996 \end{pmatrix} = \begin{pmatrix} 0.00005 \\ -0.00075 \end{pmatrix}$$

再到④, 检查  $\|\nabla f(\mathbf{x}^2)\| = \sqrt{0.00005^2 + (-0.00075)^2} = 0.00075 < \varepsilon$ , 迭代结束。最终选取  $\mathbf{x}^* = \mathbf{x}^2 = \begin{pmatrix} 0.00005 \\ -0.00075 \end{pmatrix}$ 。

共轭方向法的优缺点如下。

优点: 计算公式简单, 存储量较小, 对初始点的要求很少。对于二次函数具有二次终止性质; 收敛速度介于梯度法与牛顿法之间, 对于高维 ( $n$  较大) 的一般非线性函数有较高的效率。

缺点: 共轭方向法的收敛性强烈地依赖于精确的一维搜索, 为此将会付出很大的代价。共轭方向法的一些理论背景至今尚不清楚。例如, 周期性的重新开始, 初始搜索方向的选取, 一维搜索的精确性等, 对共轭方向法执行的影响仍有待进一步弄清。

### 11.3.6 拟牛顿法 (变尺度法)

如果将梯度法和牛顿法写成统一的形式

$$\mathbf{x}_{k+1} = \mathbf{x}_k - t_k \mathbf{H}_k \nabla f(\mathbf{x}_k)$$

则当  $\mathbf{H}_k = \mathbf{I}_k$  ( $n$  阶单位矩阵) 时, 可得到梯度计算公式  $\mathbf{x}_{k+1} = \mathbf{x}_k - t_k \mathbf{I}_k \nabla f(\mathbf{x}_k)$ ; 当  $\mathbf{H}_k = [\nabla^2 f(\mathbf{x}_k)]^{-1}$

时, 可得到牛顿法计算公式  $\mathbf{x}_{k+1} = \mathbf{x}_k - t_k [\nabla^2 f(\mathbf{x}_k)]^{-1} \nabla f(\mathbf{x}_k)$ 。

拟牛顿法的思想是, 希望能找到介于  $\mathbf{I}_k$  和  $[\nabla^2 f(\mathbf{x}_k)]^{-1}$  之间的  $\mathbf{H}_k$ , 使得  $\mathbf{H}_k$  兼有收敛速度快而又不出现  $[\nabla^2 f(\mathbf{x}_k)]^{-1}$  的迭代公式, 或希望用  $\mathbf{H}_k$  去接近  $[\nabla^2 f(\mathbf{x}_k)]^{-1}$ , 也就是说, 当  $k \rightarrow \infty$  时,  $\mathbf{H}_k \rightarrow [\nabla^2 f(\mathbf{x}_k)]^{-1}$ 。令  $\mathbf{H}_{k+1} = \mathbf{H}_k + \Delta \mathbf{H}_k$ , 式中  $\Delta \mathbf{H}_k$  为修正矩阵。随着  $\Delta \mathbf{H}_k$  的构造不同, 就可得到不同的迭代公式, 为此, 先分析  $\mathbf{H}_k \approx [\nabla^2 f(\mathbf{x}_k)]^{-1}$  的条件。

对任一个  $k$ , 若  $\nabla^2 f_k$  为正定, 则  $[\nabla^2 f_k]^{-1}$  也为正定, 所以  $s_k = -[\nabla^2 f_k]^{-1} \nabla f(\mathbf{x}_k)$  为下降方向。若将  $f(\mathbf{x})$  在  $\mathbf{x}_{k+1}$  点处进行二阶泰勒展开, 则有

$$f(\mathbf{x}) \approx f(\mathbf{x}_{k+1}) + \frac{1}{1!} \nabla f(\mathbf{x}_{k+1})^T (\mathbf{x} - \mathbf{x}_{k+1}) + \frac{1}{2!} (\mathbf{x} - \mathbf{x}_{k+1})^T \nabla^2 f(\mathbf{x}_{k+1}) (\mathbf{x} - \mathbf{x}_{k+1})$$

两边对于  $\mathbf{x}$  求导, 有  $\nabla f(\mathbf{x}) \approx \nabla f(\mathbf{x}_{k+1}) + \nabla^2 f(\mathbf{x}_{k+1})(\mathbf{x} - \mathbf{x}_{k+1})$

令  $\mathbf{x} = \mathbf{x}_k$ , 有

$$\nabla f(\mathbf{x}_k) \approx \nabla f(\mathbf{x}_{k+1}) + \nabla^2 f(\mathbf{x}_{k+1})(\mathbf{x}_k - \mathbf{x}_{k+1}) \quad (11.3.10)$$

记  $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$ ,  $\Delta \mathbf{x}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ ,  $\Delta \mathbf{g}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$ , 则式 (11.3.10) 可写为

$$\Delta \mathbf{g}_k \approx \nabla^2 f(\mathbf{x}_{k+1}) \Delta \mathbf{x}_k \quad (11.3.11)$$

若  $\nabla^2 f_k$  为正定, 则  $[\nabla^2 f_k]^{-1}$  也为正定, 因此式 (11.3.11) 可写为

$$[\nabla^2 f(\mathbf{x}_{k+1})]^{-1} \Delta \mathbf{g}_k \approx \Delta \mathbf{x}_k \quad (11.3.12)$$

式 (11.3.12) 为求  $\mathbf{H}_k$  必须满足的拟牛顿条件。归纳起来,  $\mathbf{H}_k$  必须具备以下 3 个条件。

① 对于  $\forall k$ ,  $\mathbf{H}_k$  必须正定, 从而保证  $\mathbf{s}_k = -\mathbf{H}_k^{-1} \mathbf{g}_k$  为下降方向。

② 满足拟牛顿条件式 (11.3.12):  $\mathbf{H}_{k+1} \Delta \mathbf{g}_k = \Delta \mathbf{x}_k$ 。

③ 具有易于修改的简捷形式:  $\mathbf{H}_{k+1} = \mathbf{H}_k + \Delta \mathbf{H}_k$ 。

根据这些要求, 人们从各种不同的思路去构造  $\mathbf{H}_k$ , 得到许多拟牛顿算法, 常用的有以下两种。

(1) DFP (Davidon、Fletcher 和 Powell)

$$\begin{aligned} \mathbf{H}_{k+1} &= \mathbf{H}_k + \Delta \mathbf{H}_k \\ \Delta \mathbf{H}_k &= \frac{\Delta \mathbf{x}_k \Delta \mathbf{x}_k^T}{\Delta \mathbf{x}_k^T \Delta \mathbf{g}_k} - \frac{\mathbf{H}_k \Delta \mathbf{g}_k (\mathbf{H}_k \Delta \mathbf{g}_k)^T}{\Delta \mathbf{g}_k^T \mathbf{H}_k \Delta \mathbf{g}_k} \end{aligned} \quad (11.3.13)$$

式中  $\Delta \mathbf{x}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ ,  $\Delta \mathbf{g}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$ ,  $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$ 。

(2) BFGS (Broyden、Fletcher、Goldfarb 和 Shanno)

$$\begin{aligned} \mathbf{H}_{k+1} &= \mathbf{H}_k + \Delta \mathbf{H}_k \\ \Delta \mathbf{H}_k &= \left( 1 + \frac{\Delta \mathbf{g}_k^T \mathbf{H}_k \Delta \mathbf{g}_k}{\Delta \mathbf{x}_k^T \Delta \mathbf{g}_k} \right) \frac{\Delta \mathbf{x}_k \Delta \mathbf{x}_k^T}{\Delta \mathbf{x}_k^T \Delta \mathbf{g}_k} - \frac{1}{\Delta \mathbf{x}_k^T \Delta \mathbf{g}_k} (\Delta \mathbf{x}_k \Delta \mathbf{g}_k^T \mathbf{H}_k + \mathbf{H}_k \Delta \mathbf{g}_k \Delta \mathbf{x}_k^T) \end{aligned} \quad (11.3.14)$$

式中  $\Delta \mathbf{x}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ ,  $\Delta \mathbf{g}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$ ,  $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$ 。

拟牛顿法的算法步骤如下。

① 给出初始点  $\mathbf{x}^0$ , 计算精度  $\varepsilon > 0$  和初始矩阵  $\mathbf{H}_0 = \mathbf{I}$ , 令  $k = 0$ 。

② 计算  $\mathbf{s}^k = -\mathbf{H}_k^{-1} \mathbf{g}_k$ , 沿  $\mathbf{s}^k$  进行一维搜索  $\min_{t \geq 0} f(\mathbf{x}^k + t \mathbf{s}^k)$ , 求出步长  $t_k$ , 令  $\mathbf{x}^{k+1} = \mathbf{x}^k + t_k \mathbf{s}^k$ ,

计算  $\mathbf{g}_{k+1} = \nabla f(\mathbf{x}^{k+1})$ 。

③ 如果  $\|\mathbf{g}_{k+1}\| < \varepsilon$ , 则迭代结束。取  $\mathbf{x}^* = \mathbf{x}^{k+1}$ , 否则由式 (11.3.13) 或式 (11.3.14) 求  $\mathbf{H}_{k+1}$ 。

④ 如果  $k < n-1$ , 则令  $k = k+1$ , 返回②; 如果  $k = n-1$ , 则令  $\mathbf{x}^0 = \mathbf{x}^{k+1}$ ,  $k = 0$ , 返回②。

例 11.3.8 用 DFP 拟牛顿法计算问题

$$\min f(\mathbf{x}) = \min f(x_1, x_2) = x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1$$

$$\text{取 } \mathbf{x}^0 = (1, 1)^T, \varepsilon = 0.05, \mathbf{H}_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

解 ①初值,  $\mathbf{x}^0 = (1, 1)^T, \varepsilon = 0.05, \mathbf{H}_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ , 令  $k = 0$ 。

$$\text{②计算 } \nabla f(\mathbf{x}) = \begin{pmatrix} 2x_1 - 2x_2 - 4 \\ 4x_2 - 2x_1 \end{pmatrix}, \quad \mathbf{g}_0 = \nabla f(\mathbf{x}^0) = \begin{pmatrix} -4 \\ 2 \end{pmatrix}, \quad \mathbf{s}^0 = -\mathbf{H}_0 \mathbf{g}_0 = \begin{pmatrix} 4 \\ -2 \end{pmatrix}$$

$$\mathbf{x}^0 + t\mathbf{s}^0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} + t \begin{pmatrix} 4 \\ -2 \end{pmatrix} = \begin{pmatrix} 1+4t \\ 1-2t \end{pmatrix}$$

$$\min_{t \geq 0} f(\mathbf{x}^0 + t\mathbf{s}^0) = \min_{t \geq 0} \left[ (1+4t)^2 + 2(1-2t)^2 - 2(1+4t)(1-2t) - 4(1+4t) \right]$$

$$\text{求得 } t_0 = 0.25, \quad \mathbf{x}^1 = \mathbf{x}^0 + t\mathbf{s}^0 = \begin{pmatrix} 1+4 \times 0.25 \\ 1-2 \times 0.25 \end{pmatrix} = \begin{pmatrix} 2.0 \\ 0.5 \end{pmatrix}, \quad \mathbf{g}_1 = \nabla f(\mathbf{x}^1) = \begin{pmatrix} -1 \\ -2 \end{pmatrix}.$$

③  $\|\mathbf{g}_1\| = \sqrt{(-1)^2 + (-2)^2} = \sqrt{5} > \varepsilon$ , 计算  $\mathbf{H}_1$  如下。

$$\Delta \mathbf{x}_0 = \mathbf{x}^1 - \mathbf{x}^0 = \begin{pmatrix} 1 \\ -0.5 \end{pmatrix}, \quad \Delta \mathbf{g}_0 = \mathbf{g}_1 - \mathbf{g}_0 = \begin{pmatrix} 3 \\ -4 \end{pmatrix}$$

$$\mathbf{H}_1 = \mathbf{H}_0 + \frac{\Delta \mathbf{x}_0 \Delta \mathbf{x}_0^T}{\Delta \mathbf{x}_0^T \Delta \mathbf{x}_0} - \frac{(\mathbf{H}_0 \Delta \mathbf{g}_0)(\mathbf{H}_0 \Delta \mathbf{g}_0)^T}{\Delta \mathbf{g}_0^T \mathbf{H}_0 \Delta \mathbf{g}_0} = \dots = \begin{pmatrix} 21/25 & 19/50 \\ 19/50 & 41/100 \end{pmatrix}$$

$$\text{这时 } k = 1, \text{ 返回②, 计算 } \mathbf{s}^1 = -\mathbf{H}_1 \mathbf{g}_1 = \begin{pmatrix} 8/5 \\ 6/5 \end{pmatrix}, \quad \mathbf{x}^1 + t\mathbf{s}^1 = \begin{pmatrix} 2+8t/5 \\ 0.5+6t/5 \end{pmatrix}$$

$$\min_{t \geq 0} f(\mathbf{x}^1 + t\mathbf{s}^1) = \min_{t \geq 0} \left[ (2+8t/5)^2 + 2(0.5+6t/5)^2 - 2(2+8t/5)(0.5+6t/5) - 4(2+8t/5) \right]$$

$$\text{继续③求得 } t_1 = 5/4, \quad \mathbf{x}^2 = \mathbf{x}^1 + t_1 \mathbf{s}^1 = \begin{pmatrix} 4 \\ 2 \end{pmatrix}, \quad \mathbf{g}_2 = \nabla f(\mathbf{x}^2) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \text{ 迭代结束。选取 } \mathbf{x}^* = \mathbf{x}^2 = \begin{pmatrix} 4 \\ 2 \end{pmatrix}.$$

拟牛顿法优缺点:

① 有较快的收敛速度。

② 同共轭方向法一样, 对二次函数具有二次终止的性质。

缺点: 存储量较大, 大约需要  $O(n^2)$  的存储单元, 由此对大型问题会带来不便。

## 本章小结

本章介绍了无约束函数最优化方法。

在一元函数优化计算中, 牛顿法收敛速度快, 但需要计算二阶导数。拟牛顿法用弦线段代替弧线段, 其速度比牛顿法慢, 但它只用一阶导数而不用二阶导数。黄金分割法要求函数为单峰即可, 迭代中只求函数值, 不要求导数值, 但速度比牛顿法慢。

在多元函数优化计算中, 梯度法每一轮的计算量较小, 从一般的初始点出发, 都能收敛到某个局部极小点, 但对于小的扰动是不稳定的。牛顿法收敛速度快, 为二阶收敛, 但要求  $f(\mathbf{x})$

二阶可导；在迭代中要多次计算 $[\nabla^2 f(\mathbf{x})]^{-1}$ ，计算非常困难。共轭方向法计算公式简单，存储量较小，对于二次函数具有二次终止性质；但其收敛性强烈地依赖于精确的一维搜索，为此将会付出很大的代价。拟牛顿法有较快的收敛速度，同共轭方向法一样，对二次函数具有二次终止的性质，但存储量较大，大约需要 $O(n^2)$ 的存储单元，由此对大型问题会带来不便。

## 习题 11

11.1 用一维牛顿法求解函数 $f(x) = 3x^4 - 4x^3 - 12x^2$ 的最优解，取初始点 $x_0 = -1.2$ ，迭代3次。

11.2 用一维拟牛顿法求解函数 $f(x) = 3x^4 - 4x^3 - 12x^2$ 的最优解，取初始点 $x_0 = -1.2$ ， $x_1 = -0.8$ ，迭代3次。

11.3 用黄金分割法求解函数 $f(\mathbf{x}) = 3x^4 - 4x^3 - 12x^2$ 的最优解，取初始区间 $[a_1, b_1] = [-2, 0]$ 。

11.4 计算函数 $f(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2$ 的梯度和 Hesse 矩阵。

11.5 求函数 $f(\mathbf{x}) = 100(x_2 - x_1^2) + (1 - x_1)^2$ 在点 $\mathbf{x} = (1, 1)^T$ 处的最快速度下降方向。

11.6 用梯度法求函数 $f(x) = 2x_1^2 + x_2^2$ 的最优解，取初始点 $\mathbf{x}^0 = (1, 1)^T$ ， $\varepsilon = 0.1$ 。

11.7 用牛顿法求函数 $f(x) = (x_1 - 1)^4 + x_2^2$ 的最优解，取初始点 $\mathbf{x}^0 = (0, 1)^T$ ， $\varepsilon = 0.1$ 。

11.8 证明向量 $\mathbf{x} = (1, 0)^T$ ， $\mathbf{y} = (3, -2)^T$ 关于矩阵 $\mathbf{A} = \begin{pmatrix} 2 & 3 \\ 3 & 5 \end{pmatrix}$ 共轭。

11.9 用共轭方向法求函数 $f(x) = x_1^3 + x_2^3 - 3(x_1 + x_2)$ 的最优解，取 $\mathbf{x}^0 = (0, 0)^T$ ， $\varepsilon = 0.05$ 。

11.10 用 DEP 拟牛顿法求解函数 $f(\mathbf{x}) = 2x_1^2 + x_2^2 - 4x_1 + 2$ 的最优解，取 $\mathbf{x}^0 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ ，

$$\mathbf{H}_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}。$$



## 第12章 MATLAB 编程基础及其在计算方法中的应用



### 学习要点

本章简要介绍现代数值计算软件 MATLAB 及其在计算方法中的应用，主要内容有：

(1) MATLAB 编程基础，包括编程环境和基本命令、数值与符号计算、图形显示功能、程序控制和调试。

(2) MATLAB 在计算方法中的应用，包括使用 MATLAB 编程和使用 MATLAB 内部函数两种方式实现第 3~11 章中的基本算法和部分例题。

(3) 用 C 语言实现本书的部分算法，并与 MATLAB 进行比较。



### 教学建议

现代数值计算软件 MATLAB 在计算方法中有着非常广泛的应用，要求学生能够掌握 MATLAB 的基本编程方法，并结合本章给出的约 40 个程序源代码，亲自动手编程，掌握计算方法中的基本算法。建议在实验室或多媒体教室讲解本章内容，建议学时为 4~10 学时，实验为 2~6 学时。

## 12.1 MATLAB 简介

MATLAB 由 Matrix 和 Laboratory 两词的前三个字母组合而成，原意为矩阵实验室，是 MathWorks 公司于 1984 年正式推出的数值计算软件。目前，在国际学术界，MATLAB 已经被确认为准确、可靠的科学计算标准软件。其主要特点如下。

- ① MATLAB 语言数据类型丰富，是面向矩阵（向量）计算的高级程序设计语言。
- ② MATLAB 的工作环境友好，大量引入图形用户界面和全方位帮助系统。
- ③ MATLAB 有强大的绘图功能，MATLAB 的图形显示系统具有可编辑图形窗口、支持 Tex 特殊字符集，其绘图指令简捷，读写图像文件的格式丰富。
- ④ MATLAB 的数学函数库丰富，有大量事先定义的数学函数，也有高性能数值计算的高级算法，特别适合矩阵代数领域。
- ⑤ 具有很强的符号计算能力，MATLAB 可以进行矩阵分解、微分、积分、积分变换、代数方程求解、微分方程求解等运算。
- ⑥ MATLAB 可以与外部程序进行交互，它可与其他语言编写的程序结合，具有输入/输出格式化数据的能力。
- ⑦ MATLAB 配有自己的编译器，它不但可以把全 M 函数文件编译成独立应用程序，而且也可以把 C 或 Fortran 程序与 M 文件混编成独立应用程序。
- ⑧ MATLAB 提供在多个应用领域解决难题的工具箱。

## 12.2 命令窗口和基本命令

MATLAB 是一个高度集成的编程环境，它包括命令窗口和代码编辑器。启动 MATLAB 后，即自动进入 MATLAB 工作界面，如图 12.2.1 所示，第一行为菜单栏，第二行为工具栏，下面是三个最常用的窗口。

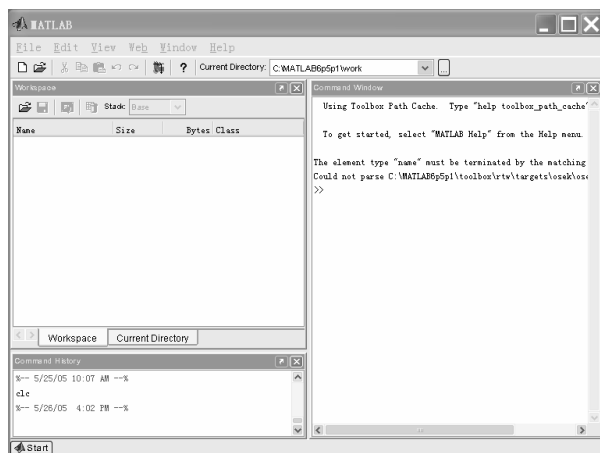


图 12.2.1 MATLAB 工作界面

左上方当前选项卡为工作空间 (Workspace)，列出内存中 MATLAB 工作空间的所有变量信息；另一个选项卡为当前目录 (Current Directory)，列出当前目录的程序文件 (.m) 和数据文件 (.mat)。左下方为命令历史 (Command History)，用右键菜单命令可以进行复制、执行和删除等操作。右边是命令窗口 Command Window。命令窗口是进行 MATLAB 操作最主要的窗口。“>>”为 MATLAB 命令提示符。在命令行中，可以用注释符 “%” 添加注释，可以用续行符 “...” 将单条语句行分成多行，可以用分隔符 “;” 将多条语句放在同一行中，命令后加 “;” 表示不显示该命令的计算结果。MATLAB 常见命令见表 12.2.1。

表 12.2.1 列出 MATLAB 的一些常见命令。

表 12.2.1 MATLAB 的一些常见命令

命 令	命 令 说 明	命 令	命 令 说 明
cd	显示或改变工作目录	hold	图形保持开关
dir	显示目录下的文件	disp	显示变量或文字内容
type	显示文件内容	path	显示搜索目录
clear	清理内存变量	save	保存内存变量到指定文件中
clf	清除图形窗口	load	加载指定文件的变量
pack	收集内存碎片，扩大内存空间	diary	日志文件命令
clc	清除目录窗口	quit	退出 MATLAB
echo	命令窗口信息显示开关	!	调用 DOS 命令
format	设置数据显示格式	size	数组尺寸
who	显示变量名	max	最大值
whos	显示变量信息	min	最小值

续表

命 令	命 令 说 明	命 令	命 令 说 明
linspace	区间等分	sum	求和
length	数组长度	reshape	重排数组
find	条件检索	input	提示输入
double	将 ASCII 码转化为数值	global	全程变量
char	将 ASCII 码转化为字符	nargin	输入变量个数
struct2cell	将结构体转化为单元	nargout	输出变量个数
cell2struct	将单元转化为结构体	tic	启动秒表
inline	内嵌函数	toc	时间读数(秒)
feval	函数求值	help	帮助
which	查找文件目录	lookfor	查找

可以直接使用 `help` 命令获得这些命令的使用说明，也可使用 `help` 命令进行分类搜索，还可以采用 `help topic` 命令形式获得具体子类的命令明细。另外，可以用 `lookfor` 命令查找包含某个关键词的所有命令。

## 12.3 变量、常量和数据类型

MATLAB 使用变量存储数据，但变量不需要声明，直接使用。常用的特殊变量为 `ans`，当最新的计算结果没有被指定给某个变量时，`ans` 就接收它。

MATLAB 中有几个特殊常量：`Pi` 表示  $\pi$ ，`Inf` 表示无穷大，`i` 或 `j` 表示虚数单位，`esp` 表示浮点运算的精度，`NaN` 表示不定值。

MATLAB 数据显示格式默认为短格式（short），实数按小数点后 4 位长度显示，数据显示格式可以使用命令 `format` 改变，也可以通过选择菜单命令 `File→Preference→Command→Format` 改变。显示格式的改变不会影响数据的实际值。

**例 12.3.1** 用不同的显示格式显示 `sqrt(2)` 的值。

显示格式	显示结果
short	1.4142
long	1.41421356237310
hex	3ff6a09e667f3bcd
bank	1041
plus	+
short e	104142e+000
long e	10414213562373095e+000
short g	104142
long g	104142135623731
rational	$\frac{1393}{985}$

MATLAB 的数据类型有标量（scalar）、向量（vector）、矩阵（matrix）、字符串（string）、单元数组（cell array）、结构体（structure）、对象（object）等。它们都以 MATLAB 数组按列存储。

单元数组用于保存不同类型、不同大小的数据，单元数组的每个元素称为单元(cell)，单元

数组允许把不同类型的 MATLAB 数组保存在不同的单元中，即单元数组的各个单元的数据类型可以不同。结构体用于保存相关的数据。它由一组称为域(field)的成员变量构成，每个域可以是不同的数据类型。与单元数组不同的是结构体把数据存于域中，而不是存于单元中。

## 12.4 数值运算

### 12.4.1 向量运算

#### (1) 向量输入

可以直接按行方式输入向量的每个元素，同一行中的元素用逗号(,)或空格符来分隔，且空格个数不限，所有元素都处于一对方括号([ ])内。

例如：

```
>> Time = [11 12 1 2 3 4 5 6 7 8 9 10]
```

显示结果：

```
Time =  
11 12 1 2 3 4 5 6 7 8 9 10
```

#### (2) 向量的加减和数乘

向量的加减要求运算的向量有相同的维数，且对应元素相加减。向量的数乘就是向量的每个元素乘以该数。

例如：

```
>>A=[1, 1, 1]; B=[8, 1, 6];  
>>A+B  
>>A-B  
>>B*2
```

显示结果：

```
A+B=  
9 2 7  
A-B=  
-7 0 -5  
B*2=  
16 2 12
```

#### (3) 向量点积

格式：

```
C = dot(A,B)
```

若 A、B 为长度相同向量，则返回向量 A 与 B 的点积。

例如：

```
>>X=[-1 0 2];Y=[-2 -1 1];  
>>Z=dot(X,Y)
```

显示结果：

```
Z =  
4
```

#### (4) 向量叉积

在数学上，两个向量的叉积是一个过两个相向量的交点且垂直于两个向量所在平面的向

量。在 MATLAB 中，用函数 cross 实现。

格式：

$C = \text{cross}(A,B)$

若 A、B 为向量，则返回 A 与 B 的叉积，即  $C=A \times B$ ，A、B 必须是含有 3 个元素的向量。

例 12.4.1 计算垂直于向量(1,2,3)和(4,5,6)的向量。

解  $\gg a=[1 \ 2 \ 3]; b=[4 \ 5 \ 6];$

$\gg c=\text{cross}(a,b)$

显示结果：

$c =$   
-3      6      -3

可得垂直于向量(1,2,3)和(4,5,6)的向量为  $\pm(-3,6,-3)$ 。

(5) 混合积

混合积由以上两个函数 dot、cross 实现。

例 12.4.2 计算向量  $a=(1,2,3)$ ， $b=(4,5,6)$  和  $c=(-3,6,-3)$  的混合积  $a \cdot (b \times c)$ 。

解  $\gg a=[1 \ 2 \ 3]; b=[4 \ 5 \ 6]; c=[-3 \ 6 \ -3];$

$\gg x=\text{dot}(a, \text{cross}(b,c))$

显示结果：

$x =$   
54

注意：先叉乘后点乘，顺序不可颠倒。

(6) 向量的范数

格式：

$n = \text{norm}(X)$

$X$  为向量，求欧几里得范数，即  $\|X\|_2 = \sqrt{\sum_k |x_k|^2}$ 。

$n = \text{norm}(X, \text{inf})$

求  $\infty$ -范数，即  $\|X\| = \max(\text{abs}(X))$

$n = \text{norm}(X, 1)$

求 1-范数，即  $\|X\|_1 = \sum_k |x_k|$

## 12.4.2 矩阵运算

(1) 矩阵输入

可以直接按行方式输入矩阵的每个元素，同一行中的元素用逗号(,)或空格符来分隔，且空格个数不限；不同的行用分号(;)分隔，所有元素都处于一对方括号([ ])内。

例如：

$\gg \text{Matrix\_B} = [1 \ 2 \ 3; 2 \ 3 \ 4; 3 \ 4 \ 5]$

显示结果：

$\text{Matrix\_B} =$  1    2    3  
              2    3    4  
              3    4    5

另外，可以用函数生成一些特殊的矩阵。

全零阵： $B = \text{zeros}(n)$       %生成  $n \times n$  全零阵

单位阵:  $Y = \text{eye}(n)$       %生成  $n \times n$  单位阵

全 1 阵:  $Y = \text{ones}(n)$       %生成  $n \times n$  全 1 阵

Hilbert 矩阵:  $H = \text{hilb}(n)$       %返回  $n$  阶 Hilbert 矩阵, 其元素为  $H(i,j)=1/(i+j-1)$ 。

**例 12.4.3** 产生一个三阶 Hilbert 矩阵。

```
>> format rat      %以有理数形式输出
```

```
>> H=hilb(3)
```

H =

1	1/2	1/3
1/2	1/3	1/4
1/3	1/4	1/5

逆 Hilbert 矩阵:  $H = \text{invhilb}(n)$       %产生  $n$  阶逆 Hilbert 矩阵

Magic(魔方)矩阵:  $M = \text{magic}(n)$       %产生  $n$  阶魔方矩阵, 该矩阵每行、每列及对角线之和相等。

例如:

```
>> M=magic(3)
```

M =

8	1	6
3	5	7
4	9	2

(2) 矩阵的加、减、乘和数乘运算

**例 12.4.4** 矩阵的加、减。

**解** >>A=[1,1,1;1,2,3;1,3,6]; B=[8,1,6;3,5,7;4,9,2]

```
>>A+B
```

```
>>A-B
```

显示结果:

A+B=

9	2	7
---	---	---

4	7	10
---	---	----

5	12	8
---	----	---

A-B=

-7	0	-5
----	---	----

-2	-3	-4
----	----	----

-3	-6	4
----	----	---

**例 12.4.5** 矩阵相乘和数乘矩阵。

**解** >>X=[2 3 4 5;1 2 2 1];

```
>>Y=[0 1 1;
```

```
1 1 0;
```

```
0 0 1;
```

```
1 0 0];
```

```
>>Z=X*Y
```

```
>>a=2*X
```

显示结果:

Z=

8	5	6
---	---	---

3	3	3
---	---	---

```
a =
4 6 8 10
2 4 4 2
```

### (3) 矩阵的除法运算和矩阵的逆运算

MATLAB 提供了两种除法运算：左除 (\) 和右除 (/)。在一般情况下， $x=a \backslash b$  是方程  $a*x=b$  的解，而  $x=b/a$  是方程  $x*a=b$  的解。可以用函数 `inv` 求矩阵的逆。

格式：

```
Y=inv(X)
```

求方阵  $X$  的逆矩阵。若  $X$  为奇异阵或近似奇异阵，将给出警告信息

#### 例 12.4.6 矩阵的除法。

```
a=[1 2 3;4 2 6;7 4 9]; b=[4;1;2];
x=a\b
```

显示结果：

```
x=
-1.5000
2.0000
0.5000
```

例 12.4.7 求  $A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 2 & 1 \\ 3 & 4 & 3 \end{pmatrix}$  的逆矩阵。

解 >>A=[1 2 3;2 2 1;3 4 3];

```
>>Y=inv(A)
```

显示结果：

```
Y =
1.0000    3.0000   -2.0000
-1.5000   -3.0000    2.5000
1.0000    1.0000   -1.0000
```

如果  $a$  为非奇异矩阵，则  $a \backslash b$  和  $b/a$  可通过  $a$  的逆矩阵与  $b$  矩阵得到

$$a \backslash b = \text{inv}(a)*b, \quad b/a = b*\text{inv}(a)$$

### (4) 矩阵转置

用运算符 “'” 可求矩阵  $A$  的转置矩阵  $A'$ ，若矩阵  $A$  的元素为实数，则与线性代数中矩阵的转置相同。若  $A$  为复数矩阵，则  $A$  转置后的元素由  $A$  对应元素的共轭复数构成。

### (5) 方阵的行列式

用函数 `det` 求  $A$  的行列式  $\det(A)$ 。

例如：

```
>> A=[1 2 3;4 5 6;7 8 9];
>> D=det(A)
```

显示结果：

```
D =
0
```

### (6) 矩阵的范数

格式：

```
n = norm(A)
```

$A$  为矩阵，求欧几里德范数  $\|A\|_2$ 。

`n = norm(A,1)`  
求  $A$  的列范数  $\|A\|_1$ 。

`n = norm(A,inf)`  
求  $A$  的行范数  $\|A\|_\infty$ 。

`n = norm(A,'fro')`  
求矩阵  $A$  的 Frobenius 范数  $\|A\|_F = \sqrt{\sum_i \sum_j |a_{ij}|^2}$ 。

(7) 矩阵的条件数

格式：

`c = cond(X)`  
求  $X$  的 2-范数的条件数。

`c = cond(X,p)`  
求  $p$ -范数的条件数， $p$  的值可以是 1、2、inf 或者 'fro'。

说明：线性方程组  $AX=b$  的条件数是一个大于或者等于 1 的实数，用来衡量关于数据中的扰动，也就是  $A$  或  $b$  对解  $X$  的灵敏度。一个差条件的方程组的条件数很大。条件数的定义为：  
 $\text{cond}(A) = \|A\| \|A^{-1}\|$ 。

## 12.5 符号运算

### 12.5.1 字符串运算

在 MATLAB 中的字符串一般是 ASCII 值的数值数组，它作为字符串表达式显示。一个字符串是由单引号括起来的简单文本。字符串中的每个字符是数组里的一个元素，因为字符串是数值数组，所以它们可以用 MATLAB 中所有可利用的数组操作工具进行操作。

MATLAB 提供了许多有用的字符串转换函数，见表 12.5.1。

表 12.5.1 字符串转换函数

字符串转换函数	说 明
<code>abs</code>	字符串转换成 ASCII 码
<code>dec2hex</code>	十进制数转换成十六进制字符串
<code>fprintf</code>	把格式化的文本写到文件中或显示在屏幕上
<code>hex2dec</code>	十六进制字符串转换成十进制数
<code>hex2num</code>	十六进制字符串转换成 IEEE 浮点数
<code>int2str</code>	整数转换成字符串
<code>lower</code>	字符串转换成小写
<code>num2str</code>	数字转换成字符串
<code>setstr</code>	ASCII 码转换成字符串
<code>sprintf</code>	用格式控制，数字转换成字符串
<code>sscanf</code>	用格式控制，字符串转换成数字
<code>str2mat</code>	字符串转换成一个文本矩阵
<code>str2num</code>	字符串转换成数字
<code>upper</code>	字符串转换成大写



MATLAB 还提供了许多字符串函数，见表 12.5.2。

表 12.5.2 字符串函数

字符串函数	说 明
eval(string)	作为一个 MATLAB 命令求字符串的值
blanks(n)	返回一个有 n 个 0 或空格的字符串
deblank	去掉字符串中后面的空格
feval	求由字符串给定的函数值
findstr	从一个字符串中找出字符串
isletter	当字符存在时返回真值
isspace	当空格字符存在时返回真值
isstr	如果输入的是一个字符串，则返回真值
lasterr	返回上一个产生 MATLAB 错误的字符串
strcmp	若字符串相同，则返回真值
strrep	用一个字符串替换另一个字符串
strtok	在一个字符串中找出第一个标记

### 12.5.2 符号表达式运算

符号表达式是代表数字、函数、算子和变量的 MATLAB 字符串或字符串数组。符号方程式是含有等号的符号表达式。无变量的符号表达式称为符号常量。符号表达式中的变量称为符号变量。当字符表达式中含有多于一个的变量时，只有一个变量是独立变量。

符号表达式可以进行如下运算。

#### (1) 提取分子和分母

如果表达式是一个有理分式（两个多项式之比），或是可以展开为有理分式（包括那些分母为 1 的分式），可利用 numden 命令来提取分子或分母。在必要时，numden 将表达式合并、有理化并返回所得的分子和分母。

例 12.5.1 提取表达式  $h = \frac{x^2+3}{2x-1} + \frac{3x}{x-1}$  的分子和分母。

```
解 >> h = '(x^2+3)/(2*x-1)+3*x/(x-1)'; %定义一个有理多项式的和式
>> [n,d]=numden(h) %有理化后提取
n=
x^3+5*x^2-3
d=
(2*x-1)*(x-1)
```

#### (2) 标准代数运算

很多标准的代数运算可以在符号表达式上执行，函数 symadd、symsub、symmul 和 symdiv 分别为加、减、乘、除表达式。

例 12.5.2 对给定两个函数  $f = 2x^2 + 3x - 5$ ,  $g = x^2 - x + 7$  进行加、减、乘、除运算。

```
解 >> f = '2*x^2+3*x-5'; %定义一个符号表达式
>> g = 'x^2-x+7';
>> symadd(f, g) %求表达式 f 与 g 的和式
ans=
3*x^2+2*x+2
```

```
>> symsub(f, g)           %求表达式 f 与 g 的差式
ans=
    x^2+4*x-12
>> symmul(f, g)           %求表达式 f 与 g 的积式
ans=
    (2*x^2+3*x-5)*(x^2-x+7)
>> symdiv(f, g)           %求表达式 f 与 g 的商式
ans=
    (2*x^2+3*x-5)/(x^2-x+7)
```

### (3) 高级运算

MATLAB 提供对符号表达式执行更高级运算的功能。函数 `compose` 用于把  $f(x)$  和  $g(x)$  复合成  $f(g(x))$ 。函数 `finverse` 用于求表达式的函数逆，如果逆不唯一，则给出警告。而函数 `symsum` 用于求表达式的符号和。

**例 12.5.3** 求给定表达式  $f(x) = \frac{1}{1+x^2}$ ,  $g = \sin(x)$  的复合函数。

```
解 >> f='1/(1+x^2)';      %定义两个符号表达式
>> g='sin(x)';
>> compose(f,g)           %求复合表达式 f(g(x))
ans=
    1/(1+sin(x)^2)
>> compose(g,f)           %求复合表达式 g(f(x))
ans=
    sin(1/(1+x^2))
```

### (4) 变换函数

函数 `sym` 可获取一个数字参量并将其转换为符号表达式。函数 `numeric` 和 `eval` 把一个符号常数（无变量符号表达式）变换为一个数值。符号函数 `sym2poly` 将符号多项式变换成它的 MATLAB 等价系数向量。函数 `poly2sym` 功能正好相反，并让用户指定用于所得结果表达式中的变量。

**例 12.5.4** 变换函数举例。

```
解 >> phi='(1+sqrt(5))/2'; %黄金分割比
>> numeric(phi)           %变换为相应的数值
ans=
    1.6180
>> eval(phi)              %完成字符串(1+sqrt(5))/2 的变换
ans=
    1.6180
>> f='2*x^2+x^3-3*x+5'; %f 是一个符号多项式
>> n=sym2poly(f)          %提取系数向量的值
n=
    1     2    -3     5
>> poly2sym(n)            %重新生成原有的符号多项式
ans=
    2*x^2+x^3-3*x+5
>> poly2sym(n, 's')       %重新生成自变量为 s 的符号多项式
ans=
    s^3+2*s^2-3*s+5
```

### (5) 符号表达式画图

在许多的场合，将表达式可视化更方便。MATLAB 提供了函数 `ezplot` 来完成该任务。

#### 例 12.5.5 表达式可视化举例。

解 >> `y=-16*x^2+64*x+96` ; %定义用于绘图的多项式

>> `ezplot(y)`

图形如图 12.5.1 所示。

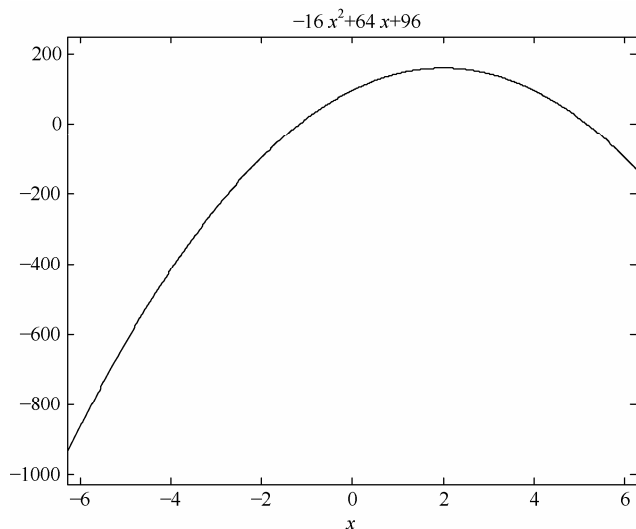


图 12.5.1 可视化表达式  $-16x^2+64x+96$  ( $-2\pi \leq x \leq 2\pi$ )

### (6) 符号表达式简化和格式化

MATLAB 用许多命令来简化或改变符号表达式。

#### 例 12.5.6 简化和格式化表达式举例。

解 >> `f=sym(' (x^2-1)*(x-2)*(x-3) ');` %定义一个函数

>> `collect(f)` %合并同类项

`ans=`

`x^4-5*x^3+5*x^2+5*x-6`

>> `honor(ans)` %变为嵌套(霍纳)形式

`ans=`

`-6+(5+(5+(-5+x)*x)*x)*x`

>> `factor(ans)` %表示为多项式的因式乘积

`ans=`

`(x-1)*(x-2)*(x-3)*(x+1)`

>> `expand(f)` %表示为各项乘积的和

`ans=`

`x^4-5*x^3+5*x^2+5*x-6`

>> `simplify(' log(2*x/y) ');` %化简多项式

`ans=`

`log(2)+log(x)-log(y)`

>> `simplify(' (-a^2+1)/(1-a) ');`

`ans=`

`a+1`

### 12.5.3 符号矩阵运算

#### (1) 符号矩阵的生成

在 MATLAB 中, 输入符号矩阵要用到符号矩阵定义函数 `sym`, 符号矩阵中的元素可以是任何的符号或者表达式, 而且长度没有限制, 要将方括号置于用于创建符号表达式的单引号中。也可以用符号定义函数 `syms` 先定义一些必要的符号变量, 再像定义普通矩阵一样输入符号矩阵。

**例 12.5.7** 输入符号矩阵举例。

```
解 >> sym_digits = sym('[1 2 3; a b c; sin(x)cos(y)tan(z)]')
sym_digits =
[1 2 3]
[a b c]
[sin(x)cos(y)tan(z)]
>> syms a b c;
>> M1 = sym('Classical');
>> M2 = sym('Jazz');
>> M3 = sym('Blues')
>> syms_matrix = [a b c; M1,M2,M3;int2str([2 3 5])]
syms_matrix =
[a b c]
[Classical Jazz Blues]
[2 3 5]
```

#### (2) 符号矩阵的运算

用函数 `symadd`、`symsub`、`symmul` 和 `symdiv` 可以对符号矩阵执行加、减、乘、除运算, 用 `sympow` 可以计算乘幂, 用 `transpose` 可以计算符号矩阵的转置, 用函数 `inverse` 和 `determ`, 可以计算符号矩阵的逆矩阵及行列式, 用函数 `charpoly` 可以求解矩阵的特征多项式, 用函数 `eigensys` 可以求解符号矩阵的特征根和特征向量。

**例 12.5.8** 符号矩阵运算举例。

```
解 >> G=sym(' [cos(t), sin(t); -sin(t), cos(t)] '); %定义一个符号矩阵
>> symadd(G,' t ') %每个元素与 t 相加
ans=
[ cos(t)+t,sin(t)+t]
[-sin(t)+t,cos(t)+t]
>> G1=symmul(G,G) %G 的乘积, 也可用函数 sympow(G,2)
G1=
[cos(t)^2-sin(t)^2, 2*cos(t)*sin(t)]
[-2*cos(t)* sin(t), cos(t)^2-sin(t)^2]
>> simple(G1) %化简
G1=
[cos(2*t), sin(2*t)]
[-sin(2*t),cos(2*t)]
>> H=sym(hilb(3)); %定义三阶希尔伯特符号矩阵
>> determ(H) %求 H 的行列式
ans=
1/2160
```

```

>> J=inverse(H) %求 H 的逆
J=
     9,     -36,     30
    [-36,     192,    -180]
     30,    -180,     180]
>> determ(J) %求逆的行列式
ans=
    2160
>> F=sym(' [1/2,1/4;1/4,1/2] '); %定义符号矩阵
>> eigensys(F) %求 F 的特征值
ans=
     [3/4]
     [1/4]
>> [V,E]=eigensys(F) %求 F 的特征值 E 和特征向量 V
V=
    [-1,1]
    [ 1,1]
E=
    [1/4]
    [3/4]

```

## 12.5.4 符号微积分运算

### (1) 求极限

格式说明如下：

```
limit(F,x,a)
```

计算符号表达式  $F=F(x)$  的极限值，当  $x \rightarrow a$  时。

```
limit(F,a)
```

用命令 `findsym(F)` 确定  $F$  中的自变量，设为变量  $x$ ，再计算  $F$  的极限值，当  $x \rightarrow a$  时。

```
limit(F)
```

用命令 `findsym(F)` 确定  $F$  中的自变量，设为变量  $x$ ，再计算  $F$  的极限值，当  $x \rightarrow 0$  时。

```
limit(F,x,a,'right')或 limit(F,x,a,'left')
```

计算符号函数  $F$  的单侧极限：左极限  $x \rightarrow a^-$  或右极限  $x \rightarrow a^+$ 。

**例 12.5.9** 求极限运算举例。

**解** >>syms x a t h n;

```
>>L1 = limit((cos(x) -1)/x)
```

```
>>L2 = limit(1/x^2,x,0,'right')
```

```
>>L3 = limit(1/x,x,0,'left')
```

```
>>L4 = limit((log(x+h) -log(x))/h,h,0)
```

```
>>v = [(1+a/x)^x, exp(-x)];
```

```
>>L5 = limit(v,x,inf,'left')
```

```
>>L6 = limit((1+2/n)^(3*n),n,inf)
```

计算结果为：

```
L1 =
```

```
0
```

```
L2 =
```

```

inf
L3 =
-inf
L4 =
1/x
L5 =
[ exp(a),      0]
L6 =
exp(6)

```

## (2) 符号函数的微分

微分函数 `diff(S)` 有四种格式:

`diff(S,'v')` 或 `diff(S,sym('v'))`

对表达式 `S` 中指定符号变量 `v` 计算 `S` 的 1 阶导数。

`diff(S)`

对表达式 `S` 中的符号变量 `v` 计算 `S` 的 1 阶导数, 其中 `v=findsym(S)`。

`diff(S,n)`

对表达式 `S` 中的符号变量 `v` 计算 `S` 的 `n` 阶导数, 其中 `v=findsym(S)`。

`diff(S,'v',n)`

对表达式 `S` 中指定的符号变量 `v` 计算 `S` 的 `n` 阶导数。

### 例 12.5.10 微分举例。

```

解 >> f='a*x^3+x^2-b*x-c';           %定义一个符号表达式
    >> diff(f)                         %求默认变量 x 的微分
    ans=
        3*a*x^2+2*x-b
    >> diff(f,'a ')                   %求变量 a 的微分
    ans=
        x^3
    >> diff(f,2)                       %求变量 x 的二阶微分
    ans=
        6*a*x+2
    >> diff(f,'a ',2)                 %求变量 a 的二阶微分
    ans=
        0

```

## (3) 符号函数的积分

积分函数 `int(S)`, 其中 `S` 是一个符号表达式, 它力图求出另一符号表达式 `F` 使 `diff(F)=f`。同微分一样, 积分函数有多种格式。

格式说明如下:

`R = int(S,v)`

对符号表达式 `S` 中指定的符号变量 `v` 计算不定积分。要注意的是, 表达式 `R` 只是函数 `S` 的一个原函数, 后面没有带任意常数 `C`。

`R = int(S)`

对符号表达式 `S` 中的符号变量 `v` 计算不定积分, 其中 `v=findsym(S)`。

`R = int(S,v,a,b)`

对表达式 `s` 中指定的符号变量 `v` 计算从 `a` 到 `b` 的定积分。

`R = int(S,a,b)`

对符号表达式  $s$  中的符号变量  $v$  计算从  $a$  到  $b$  的定积分, 其中  $v=\text{findsym}(S)$ 。

### 例 12.5.11 积分举例。

```
解 >> f='sin(s+2*x)';           %定义一个符号函数
    >> int(f)                     %求变量 x 的积分
    ans=
        -1/2*cos(s+2*x)
    >> int(f,'s')                 %求变量 s 的积分
    ans=
        -cos(s+2*x)
    >> int(f,pi/2,pi)            %求变量 x 的从  $\pi/2$  到  $\pi$  的定积分
    ans=
        -cos(x)
    >> int(f,'s',pi/2,pi)        %求变量 s 的从  $\pi/2$  到  $\pi$  的定积分
    ans=
        cos(2*x) - sin(2*x)
    >> int(f,'m','n')            %求变量 x 的从 m 到 n 的定积分
    ans=
        -1/2*cos(s+2*n)+1/2*cos(s+2*m)
```

## 12.5.5 方程求解

### (1) 求解单个代数方程

MATLAB 还提供了求解符号表达式的工具。如果表达式不是一个方程式 (不含等号), 则在求解之前函数 `solve` 将表达式置成等于 0。

### 例 12.5.12 求解方程举例。

```
解 >> solve('a*x^2+b*x+c')      %求 2 次方程的根
    ans=
        [1/2/a*(-b+(b^2-4*a*c)^1/2)]
        [1/2/a*(-b-(b^2-4*a*c)^1/2)]
```

结果是符号向量, 其元素是方程的 2 个解。

### (2) 求解单个微分方程

MATLAB 中的函数 `dsolve` 可以计算常微分方程的符号解。用字母  $D$  来表示求微分,  $D2, D3, \dots$  表示重复求微分, 并以此来设定方程。任何  $D$  后所跟的字母为因变量。方程  $\frac{d^2 y}{dx^2}=0$  用符号表达式  $D2y=0$  来表示。独立变量可以指定或由 `symvar` 规则选定为默认。

### 例 12.5.13 求一阶微分方程 $\frac{dy}{dx}=1+y^2$ 的通解。

```
解 >> dsolve('Dy=1+y^2')        %求通解
    ans=
        -tan(-x+C1)
```

其中,  $C1$  是积分常数。

表 12.5.3 至表 12.5.8 列出了常用的符号运算函数。

表 12.5.3 符号表达式的运算函数

函 数	意 义
numeric	符号到数值的转换
pretty	显示悦目的符号输出
subs	替代子表达式
sym	建立符号矩阵或表达式
symadd	符号加法
symdiv	符号除法
symmul	符号乘法
symop	符号运算
sympow	符号表达式的幂运算
symrat	有理近似
symsub	符号减法
symvar	求符号变量

表 12.5.4 符号表达式的简化函数

函 数	意 义
collect	合并同类项
expand	展开
factor	因式
simple	求解最简形式
simplify	简化
symsum	和级数

表 12.5.5 符号多项式函数

函 数	意 义
charpoly	特征多项式
horner	嵌套多项式表示
numden	分子或分母的提取
poly2sym	多项式向量到符号的转换
sym2poly	符号到多项式向量的转换

表 12.5.6 符号微积分函数

函 数	意 义
diff	微分
int	积分
jordan	约当标准形
taylor	泰勒级数展开

表 12.5.7 求解符号方程函数

函 数	意 义
compose	函数的复合
dsolve	微分方程的求解
finverse	函数逆
linsolve	齐次线性方程组的求解
solve	代数方程的求解



表 12.5.8 符号线性代数函数

函 数	意 义
charpoly	特征多项式
determ	矩阵行列式的值
eigensys	特征值和特征向量
inverse	矩阵逆
jordan	约当标准形
linsolve	齐次线性方程组的解
transpose	矩阵的转置

## 12.6 图形可视化

MATLAB 可以表示出数据的二维、三维，甚至四维的图形。通过对图形的线型、立面、色彩、光线、视角等属性的控制，可把数据的内在特征表现得淋漓尽致。下面简单介绍图形的基本命令。

### 12.6.1 二维图形绘制

命令: plot

用法:

plot(X,Y)若 X,Y 均为实数向量，且为同维向量（可以不是同型向量）， $X=[x(i)]$ ， $Y=[y(i)]$ ，则 plot(X,Y)先描出点 $(x(i),y(i))$ ，然后用直线依次相连。

plot(Y)若 Y 为实数向量，Y 的维数为 m，则 plot(Y)等价于 plot(X,Y)，其中  $X=1:m$ 。

plot(X1,Y1,X2,Y2...)，其中  $X_i$  与  $Y_i$  成对出现，plot(X1,Y1,X2,Y2...)将分别按顺序取两个数据  $X_i$  与  $Y_i$  进行画图。

plot(X1,Y1,LineStyle1,X2,Y2,LineStyle2...) 将按顺序分别画出由三个参数  $X_i,Y_i,LineSpec_i$  定义的线条。其中，参数 LineSpec<sub>i</sub> 指明线条的类型、标记符号和画线用的颜色，见表 12.6.1~12.6.3。

表 12.6.1 线型

定义符	-	--	:	-.
线型	实线（默认值）	画线	点线	点画线

表 12.6.2 标记符号

定义符	+	o(字母)	*	.	x	d	^
标记类型	加号	小圆圈	星号	实点	交叉号	棱形	向上三角形
定义符	v	>	<	s	h	P	
标记类型	向下三角形	向右三角形	向左三角形	正方形	正六角星	正五角星	

表 12.6.3 颜色

定义符	r (red)	g (green)	b (blue)	c(cyan)	m (magenta)	y (yellow)	k (black)	w (white)
颜色	红色	绿色	兰色	青色	品红	黄色	黑色	白色

上述属性的定义可用字符串来定义。

**例 12.6.1** 绘制二维曲线。

解 >>t = 0:pi/20:2\*pi;  
>>plot(t,t.\*cos(t),'-r\*')  
>>hold on  
>>plot(exp(t/100).\*sin(t-pi/2),'--mo')  
>>plot(sin(t-pi),'bs')  
>>hold off

图形结果如图 12.6.1 所示。

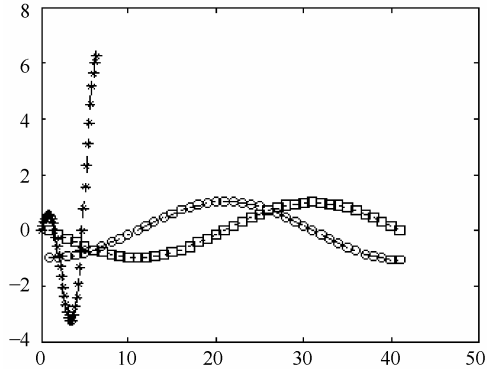


图 12.6.1 二维曲线图

MATLAB 除了直接绘制二维图形函数外，还提供了各种分析结果的可视化功能函数，这些函数操作简单，使用方便，见表 12.6.4。

表 12.6.4 MATLAB 图形函数

函数名	绘图功能	函数名	绘图功能	函数名	绘图功能
area	填充面积图	fill	填充多边形	ribbon	三维带形图
bar	条形图	fplot	函数图	stem	火柴杆图
barh	水平柱图	rose	扇形统计图	stairs	台阶图
comet	慧星形轨图	hist	直方统计图	contour	等高线图
errorbar	误差条形图	pareto	Pareto 图	clabel	等高线图仰角标签
ezplot	函数图	pie	圆饼图	pcolor	伪色图
feather	箭头(矢量)图	plotmatrix	矩阵点图	quiver	场图

**12.6.2 三维图形绘制**

命令: plot3

功能: 将绘制二维图形的函数 plot 的特性扩展到三维空间。

用法: 函数格式除了包括第三维的信息 (比如 Z 方向) 之外，其他与二维函数 plot 相同。

plot3 一般语法调用格式是 plot3(x<sub>1</sub>,y<sub>1</sub>,z<sub>1</sub>,S<sub>1</sub>,x<sub>2</sub>,y<sub>2</sub>,z<sub>2</sub>,S<sub>2</sub>…), 这里 x<sub>n</sub>,y<sub>n</sub> 和 z<sub>n</sub> 是向量或矩阵, S<sub>n</sub> 是可选的字符串, 用来指定颜色、标记符号或线形。

例 12.6.2 绘制三维螺旋线。

```
解 >> t=0:pi/50:10*pi;
    >> plot3(sin(t),cos(t),t)
    >> title('Helix'),xlabel('sint(t)'),ylabel('cos(t)'),zlabel('t')
    >> text(0,0,0, 'Origin')
    >> grid
    >> v = axis
    v =
        -1     1    -1     1     0    40
```

输出如图 12.6.2 所示。

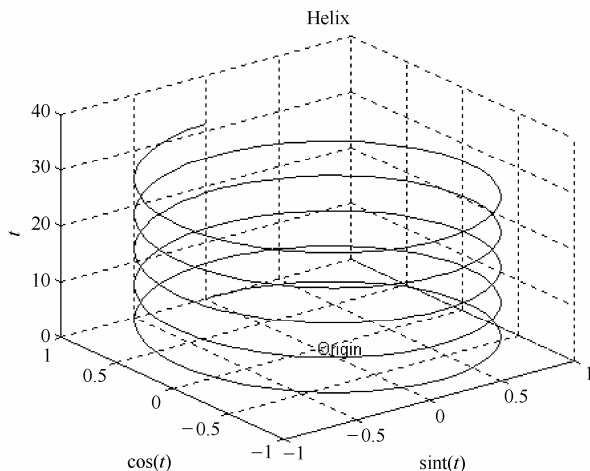


图 12.6.2 螺旋线图

从例 12.6.2 可看出，二维图形的所有基本特性在三维图形中仍然存在。`axis` 命令扩展到三维只是返回  $Z$  轴界限（0 和 40），在数轴向量中增加两个元素。函数 `zlabel` 用来指定  $Z$  轴的数据名称。函数 `grid` 在图底部绘制三维网格。函数 `text(x,y,z, 'string')` 在由三维坐标  $x,y,z$  所指定的位置放一个字符串。

MATLAB 可以对三维图形进行消隐、裁剪、缩放、色彩的调制和渲染以及光照等处理。

## 12.7 程序设计

### 12.7.1 命令文件与函数文件

命令文件也称脚本文件，它是命令的简单叠加。MATLAB 会自动按顺序执行文件中的命令。命令文件在运行过程中可以调用 MATLAB 工作域内的所有数据，所产生的变量均为全局变量而且不需要预先定义。

函数文件与命令文件类似之处在于，函数文件都是有 `.m` 扩展名的文本文件。函数文件不进入命令窗口，它是由文本编辑器所创建的外部文本文件。函数文件的函数名和文件名必须相同。函数文件以 `function` 开头，格式为：

```
function 输出变量=函数名(输入变量)
    语句
end
```

MATLAB 第一次执行一个函数文件时，它打开相应的文本文件并将文件中的命令编辑成存储器的内部表示，以加速执行以后所有的调用。函数可以有零个或多个输入参量，也可以有零个或多个输出参量。`return` 命令提供了一种结束函数的简单方法。

**例 12.7.1** 计算第  $n$  个 Fibonacci 数。

**解** 打开 MATLAB 编辑窗口，编写如下程序：

```
function f=fibfun(n)
If n>2
f=fibfun(n-1)+fibfun(n-2)
else
f=1;
end
```

编写完毕，以 `fibfun` 为文件名存盘，然后在 MATLAB 命令窗口中执行如下命令：

```
>>fibfun(17)
ans=
1597
```

结果为第 17 个 Fibonacci 数。

## 12.7.2 控制语句

MATLAB 提供三种控制流结构：`for` 循环、`while` 循环和 `if-else-end` 结构。由于这些结构经常包含大量的 MATLAB 命令，故经常出现在函数文件中，而不是直接加在 MATLAB 提示符下。

(1) `for` 循环

`for` 循环允许一组命令以预定的且固定的次数重复。`for` 循环的一般格式为：

```
for x=array
{commands}
end
```

在 `for` 和 `end` 语句之间的 `{commands}` 按数组中的每列执行一次。在每次迭代中，`x` 被指定为数组的下一列，即在第  $n$  次循环中，`x=array(:, n)`。

**例 12.7.2** `for` 循环举例。

**解** `>> for n=1:10`

```
    x(n)=sin(n*pi/10);
end
>> x
x =
Columns 1 through 7
    0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090
Columns 8 through 10
    0.5878    0.3090    0.0000
```

**注意：**如果有一个等效的数组方法来求解给定的问题，应避免使用 `for` 循环。例 12.7.2 可被重写为：

```
>> n=1:10;
>> x=sin(n*pi/10)
x =
Columns 1 through 7
    0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090
```

Columns 8 through 10

0.5878    0.3090    0.0000

两种方法得出同样的结果，而后者执行更快，更直观，要求较少的输入。

## (2) while 循环

与 for 循环以固定次数求一组命令的值相反，while 循环以不确定的次数求一组语句的值。

while 循环的一般格式为：

```
while expression
{commands}
end
```

只要在表达式 expression 中的所有元素为真，就执行 while 和 end 语句之间的 {commands}。通常，表达式的求值给出一个标量值，但数组值也同样有效。在数组值情况下，所得到数组的所有元素必须都为真。

**例 12.7.3** Fibonacci 数组的元素满足 Fibonacci 规则： $a_{k+2} = a_k + a_{k+1}$  ( $k = 1, 2, \dots$ )，且  $a_1 = a_2 = 1$ ，求该数组中第一个大于 10000 的元素。

**解** >>a(1)=1;a(2)=1;i=2;  
>>while a(i)<=10000  
    a(i+1)=a(i-1)+a(i);      %当现有的元素仍小于 10000 时，求解下一个元素  
    i=i+1;  
end;  
>>i,a(i),  
i =  
    21  
ans =  
    10946

## (3) if-else-end 结构

在很多情况下，命令的序列必须根据关系的检验有条件地执行。在编程语言里，这种逻辑由某种 if-else-end 结构来提供。最简单的 if-else-end 结构为：

```
if expression
{commands}
end
```

如果在表达式 expression 中的所有元素为真（非零），那么就执行 if 和 end 语言之间的 {commands}。

有两个选择的 if-else-end 结构为：

```
if expression
    commands evaluated if true
else
    commands evaluated if false
end
```

在这里，如果表达式 expression 为真，则执行第一组命令；如果表达式 expression 为假，则执行第二组命令。

**例 12.7.4** 用 for 循环命令来求 Fibonacci 数组中第一个大于 10000 的元素。

**解** >>n=100;a=ones(1,n);  
>>a(1)=1;a(2)=1;  
>>for i=3:n

```

a(i)=a(i-1)+a(i-2);
if a(i)>=10000
    a(i),
    break; %跳出所在的一级循环
end;
end;
ans =
    10946
>>i
i =
    21

```

除了三种基本控制流结构外，MATLAB 还提供了多重分支结构 `switch`，用于对异常进行处理。置于 `for`（或 `while`）循环内的语句 `try...catch`，根据条件执行的 `continue`、`break`、`pause` 语句，以及函数返回语句 `return`。这些语句都与其他编程语言，如 C++ 语言等，相同或相似。编程时可通过 `help` 帮助功能进行查阅，十分方便。

### 12.7.3 调试方法

MATLAB 提供的调试器可以帮助分析程序的运行和变量值的变化，查找程序的错误。调试中遇到的错误一般分成三类。

- ① 语法错误：不符合 MATLAB 的语法规则、错误的关键字等。
- ② 运行时错误：当一条语句力图执行一个不能执行的操作时，就会发生运行的错误。
- ③ 逻辑错误：当程序未按预期方式执行时，就会产生逻辑错误，即产生不正确的结果。

程序调试的任务就是，找出并改正语法错误和运行时错误使程序正常运行，若运行结果不对，则确定导致错误结果的原因，以及错误发生的地方。

MATLAB 调试器功能通过 `Debug` 菜单和 `Breakpoints` 菜单中的命令实现。

`Debug` 菜单命令：

<code>Step</code>	逐语句执行程序代码的下一条可执行语句，但不跟踪到过程中
<code>Step In</code>	逐语句执行程序代码的下一条可执行语句，并跟踪到过程中
<code>Step Out</code>	执行当前过程的其他部分，并在调用过程的下一行处中断执行
<code>Run</code>	运行程序
<code>Continue</code>	继续运行直到程序结束
<code>Go Until Cursor</code>	运行到光标处
<code>Exit Debug Mode</code>	退出调试模式

`Breakpoints` 菜单命令：

<code>Set→Clear Breakpoints</code>	设置，删除断点
<code>Clear All Breakpoints</code>	清除所有断点
<code>Stop If Error</code>	如果有错误，就停止
<code>Stop If Warning</code>	如果有警告，就停止
<code>Stop If NaN Or Inf</code>	如果计算出现 NaN（不是一个数）或 Inf（无穷大），就停止
<code>Stop If All Error</code>	只要有错误，就停止

调试中的常用方法如下。

### (1) 设置删除断点的方法

在代码编辑窗口中，先将光标定位到某一条可执行语句上，然后选择菜单命令 **Breakpoints→Set→Clear Breakpoints**（或按 F12 键），即可在此行设置/删除断点；或者，直接单击要设置删除断点的语句左边的“-”符号（“-”表示该行为可执行语句），设置断点。设置了断点后，“-”变为红色的实心圆（断点指示器）；删除断点后，红色的实心圆又变为“-”。

### (2) 在断点处检查程序的方法

程序运行到断点处并被中止后，可以检查程序的当前状态（观察已执行语句中变量的数值）。具体方法是，将光标移到变量名上，光标处显示该变量的当前值。或者，将变量（或表达式）复制到工作环境中执行，也可以观察到当前该变量（或表达式）的当前值。如果要观察设有断点的语句在运行时发生了什么，就必须至少再运行一条语句。为此要使用跟踪或单步运行。

### (3) 运行程序的选定部分

如果能够识别产生错误的语句，那么单个断点有助于对问题的定位。但更常见的情况是，只知道产生错误的代码的大体区域。这时，先通过设置断点将问题区域进行隔离，然后用跟踪和单步执行来观察每条语句的效果。

### (4) 单步执行

可用 **Debug** 菜单中的 **Step** 或 **Step In** 命令来一次一条语句地执行代码，这也称为单步执行。在单步通过每条语句之后，可以通过查看有关变量的数值变化来判断是否存在错误。

单步执行代码的步骤如下。

① 先在需要停止执行的地方设置一个断点。如果希望从程序开头逐行调试，可在程序第一条可执行语句处设置断点。

② 选择菜单命令 **Debug→Step**（或 **Step In**）命令，每执行一次，即往下执行一条语句。

当程序处在中断模式时，可用 **Go Until Cursor**（运行到光标处）功能在代码的后部选择想要停止运行的语句，这样可以略过不感兴趣的那部分代码，例如巨大的循环。

运行到光标处的步骤如下。

① 把程序设置为中断模式。

② 把光标设置在需要停止运行的地方。

③ 选择菜单命令 **Debug→Go Until Cursor**。

## 12.8 MATLAB 在计算方法中的应用

MATLAB 可以像其他计算机语言（如 C 语言）一样，通过编程实现本书中的所有算法，而且，更重要的是，MATLAB 具有许多经过优化的数值计算函数，因此，MATLAB 在计算方法中具有非常广泛的应用。本节从使用 MATLAB 编程和利用 MATLAB 内部的数值计算函数两个方面，实现本教材第 3~11 章的基本算法和部分例题。另外，选择一些典型算法，编写了 C 语言程序，与相应的 MATLAB 程序进行比较。

### 12.8.1 方程求根

#### 1. 二分法

首先编写 MATLAB 函数文件 `agui_bisect.m`：

```

function x=agui_bisect(fname,a,b,e)
% fname 为函数名, a,b 为区间端点, e 为精度
fa=feval(fname,a);
fb=feval(fname,b);
if fa*fb>0 error('两端函数值为同号');end
k=0
x=(a+b)/2
while (b-a)>(2*e)
    fx=feval(fname,x);
    if fa*fx<0
        b=x;
        fb=fx;
    else
        a=x;
        fa=fx;
    end
    k=k+1
    x=(a+b)/2
end

```

例 12.8.1 在 MATLAB 命令窗口中求解例 3.2.1。

解 >> fun=inline('sin(x)-x\*x/4')  
 >> x=agui\_bisect(fun,1.5,2,1e-2)

计算结果为:

k	x
0	1.75000000000000
1	1.87500000000000
2	1.93750000000000
3	1.90625000000000
4	1.92187500000000
5	1.92968750000000

## 2. 牛顿迭代法

首先编写 MATLAB 函数文件 agui\_newton.m:

```

function x=agui_newton(fname,dfname,x0,e)
% fname 为函数名, dfname 为函数 fname 的导函数, x0 为迭代初值
%e 为精度, N 为最大迭代次数(默认为 100)
N=100;
x=x0;
x0=x+2*e;
k=0;
while abs(x0-x)>e&k<N
    k=k+1
    x0=x;
    x=x0-feval(fname,x0)/feval(dfname,x0);
    disp(x)
end

```



```
end
if k==N warning('已达最大迭代次数'); end
```

例 12.8.2 在 MATLAB 命令窗口中求解例 3.4.1。

解 >> fun=inline('x^3-x-1')

```
fun =      Inline function:
      fun(x) = x^3-x-1
>> dfun=inline('3*x^2-1')
dfun =      Inline function:
      dfun(x) = 3*x^2-1
>> x=agui_newton(fun,dfun,1.5,1e-6)
```

计算结果为：

k	X
1	1.34782608695652
2	1.32520039895091
3	1.32471817399905
4	1.32471795724479

例 12.8.3 用牛顿迭代公式  $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$  求方程  $x - e^{-x} = 0$  在 0.5 附近的根。

解 在 MATLAB 命令窗口中输入：

```
>> f=inline('x-exp(-x)')
f =
      Inline function:
      f(x) = x-exp(-x)
>> df=inline('1+exp(-x)')
df =
      Inline function:
      df(x) = 1+exp(-x)
>> x=agui_newton(f,df,0.5,1e-6)
```

计算结果为：

k	x
1	0.56631100319722
2	0.56714316503486
3	0.56714329040978

例 12.8.4 用 C 语言实现例 12.8.3 中求方程  $x - e^{-x} = 0$  在 0.5 附近的根。

解 首先编写牛顿迭代法的 C 语言程序如下：

```
# include "stdio.h"
# include "math.h"
float f(float x){
    float y;
    y=x-exp(-x);
    return y;
}
float df(float x) {
    float y;
    y=1+exp(-x);
```

```

        return y;
    }
    main(){
        int N,k;
        float x0,eps,x1;
        printf("input x0,eps,N:");
        scanf("%f,%f,%d",&x0,&eps,&N);
        for(k=1;k<=N;k++){
            if(f(x0)==0) { printf("\nthe result is %f\n",x0);break; }
            x1=x0-f(x0)/df(x0);
            printf("\nx%d= %f\n",k,x1);
            if(fabs(x1-x0)<eps) { printf("\nthe result is %f\n k=%d",x1,k);break;}
            x0=x1;
        }
        if(k>N) printf("Failed!!");
    }
}

```

输入初值:

input x0,eps,N:0.5,1e-6,10

结果为:

```

x1= 0.566311
x2= 0.567143
x3= 0.567143
the result is 0.567143
k=3

```

Press any key to continue

从计算结果可以看出，在精度为 0.0000001 的前提下，用两种语言实现牛顿迭代法求解方程  $x - e^{-x} = 0$ ，都需要迭代 3 次。不过 MATLAB 表示的浮点数的位数较长，并且，精度以外的数位与下面用 MATLAB 内部函数计算的结果几乎完全一致。

### 3. 与方程求根有关的 MATLAB 数值计算函数

#### (1) 求一元方程实根函数 fzero

格式说明如下:

```
x = fzero(fun,x0)
```

求方程  $\text{fun}=0$  在  $x_0$  附近的根，默认精度为  $\text{eps}$ 。

```
x = fzero(fun,[a,b])
```

求方程  $\text{fun}=0$  在  $[a,b]$  内的根，默认精度为  $\text{eps}$ 。其中， $\text{fun}(a)*\text{fun}(b)<0$ 。

**例 12.8.5** 求方程  $x - e^{-x} = 0$  在区间  $[0,1]$  内的根。

**解** 在 MATLAB 命令窗口中输入:

```

>> f='x-exp(-x)'
f =
x-exp(-x)
>> fzero(f,[0,1])
ans =
0.56714329040978

```

计算结果与例 12.8.3 基本相同。

(2) 求多项式的所有复根函数 roots

格式说明如下:

```
x=roots(p)
```

求多项式的所有复根,  $p$  为多项式的系数。

**例 12.8.6** 求方程  $f(x) = x^3 - x - 1 = 0$  的根。

**解** >> roots([1 0 -1 -1])

```
ans =  
    1.32471795724475  
   -0.66235897862237 + 0.56227951206230i  
   -0.66235897862237 - 0.56227951206230i
```

计算结果与例 12.8.2 基本相同。

(3) 求一元函数的根函数 fzero

格式说明如下:

```
x=fzero(fun,x0)
```

返回一元函数 fun 在  $x_0$  附近的根。

**例 12.8.7** 求方程  $f(x) = x^3 - x - 1 = 0$  的实根。

**解** >> f=inline('x^3-x-1')

```
f=Inline function:  
    f(x) = x^3-x-1  
>> x=fzero(f,1.5)
```

```
x =  
    1.32471795724475
```

计算结果与例 12.8.2 基本相同。

(4) 求一元或多元函数的根函数

格式说明如下:

```
[x,f,h]=fsolve(fun,x0)
```

返回一元或多元函数 fun 在  $x_0$  附近的根。 $h > 0$  表示可靠, 否则不可靠。

**例 12.8.8** 求方程  $f(x) = x^3 - x - 1 = 0$  的实根。

**解** >> [x,f,h]=fsolve(f,1.5)

```
x =  
    1.32471795724479  
f =  
    2.016165012719284e-013  
h =  
    1
```

计算结果与例 12.8.2 基本相同。

## 12.8.2 解线性方程组的直接法

### 1. 列主元高斯消去法

首先编写 MATLAB 函数文件 agui\_gauss.m:

```
function x=agui_gauss(a,b)  
%列主元高斯消去法解方程组 ax=b
```

```

n=length(b);
a=[a,b];
for k=1:(n-1)
    %选主元
    [ar,r]=max(abs(a(k:n,k)));
    r=r+k-1;
    if r>k
        t=a(k,:);a(k,:)=a(r,:);a(r,:)=t;
    end
    %消元
    a((k+1):n,(k+1):(n+1))=a((k+1):n,(k+1):(n+1))-a((k+1):n,k)/a(k,k)*a(k,(k+1):(n+1));
    a((k+1):n,k)=zeros(n-k,1);
    a
end
%回代
x=zeros(n,1);
x(n)=a(n,n+1)/a(n,n);
for k=n-1:-1:1
    x(k,:)=(a(k,n+1)-a(k,(k+1):n)*x((k+1):n))/a(k,k);
end

```

例 12.8.9 在 MATLAB 命令窗口中求解例 4.2.5。

```

解 >> a=[12 -3 3;-18 3 -1;1 1 1]
a =
    12    -3     3
   -18     3    -1
     1     1     1
>> b=[15;-15;6]
b =
    15
   -15
     6
>> x=agui_gauss(a,b)
a =
   -18.0000     3.0000    -1.0000   -15.0000
         0    -1.0000     2.3333     5.0000
         0     1.1667     0.9444     5.1667
a =
   -18.0000     3.0000    -1.0000   -15.0000
         0     1.1667     0.9444     5.1667
         0         0     3.1429     9.4286
x =
    1.0000
    2.0000
    3.0000

```

计算结果与例 4.2.5 相同。

例 12.8.10 用 C 语言求解例 4.2.5。

解 首先编写列主元高斯消去法的 C 语言程序：

```

#include <math.h>
#include<stdio.h>
#include <stdlib.h>
#include <conio.h>
#define n 3
#define eps 1e-6
main (){
    int i,j,k,r;
    double c,a[n][n+1]={ {12, -3,3,15},{-18,3, -1, -15},{1,1,1,6}};
    for(k=0;k<n-1;k++){
        r=k;
        for(i=k;i<n;i++) if (fabs(a[i][k])>fabs(a[r][k]))r=i;
        if(fabs(a[r][k])<eps) { printf("\n 消元失败");exit(0); }
        if(r>k){
            for(j=k;j<n+1;j++) { c=a[k][j];a[k][j]=a[r][j];a[r][j]=c; }
        }
        for(i=k+1;i<n;i++){
            a[i][k]=a[i][k]/a[k][k];
            for(j=k+1;j<n+1;j++) a[i][j]=a[i][j] -a[i][k]*a[k][j];
        }
    }
    a[n-1][n]=a[n-1][n]/a[n-1][n-1];
    for(k=n-2;k>=0;k--) {
        c=0;
        for(j=k+1;j<n+1;j++)
            c=c+a[k][j]*a[j][n];
        a[k][n]=(a[k][n] -c)/a[k][k];
    }
    for(k=0;k<n;k++) printf("\na[%d]=%12.8f  \n",k,a[k][n]);
}

```

计算结果为:

```

this is the result:
a[0]=  1.00000000
a[1]=  2.00000000
a[2]=  3.00000000
Press any key to continue

```

用 C 和 MATLAB 两种语言实现高斯列主元消去法, 计算结果基本相同, 但是在实现手法上, MATLAB 语言明显优于 C 语言, 这主要是因为 MATLAB 适合矩阵计算, 在算法中进行选主元、消元和回代, 都非常方便。

## 2. 矩阵的直接三角分解法

首先编写 MATLAB 函数文件 agui\_lu.m:

```

function [l,u,y,x]=agui_lu(a,b)
%求可逆矩阵 a 的 Doolittle 分解, l 返回单位下三角矩阵, u 返回上三角矩阵
n=length(a);
u=zeros(n,n);

```

```

l=eye(n,n);
u(1,:)=a(1,:);
l(2:n,1)=a(2:n,1)/u(1,1);
for k=2:n
    u(k,k:n)=a(k,k:n)-l(k,1:k-1)*u(1:k-1,k:n);
    l(k+1:n,k)=(a(k+1:n,k)-l(k+1:n,1:k-1)*u(1:k-1,k))/u(k,k);
end
l
u
%解 Ly=b
y=zeros(n,1);
y(1)=b(1);
for k=2:n
    y(k)=b(k)-l(k,1:k-1)*y(1:k-1);
end
y
%解 Ux=y
x=zeros(n,1);
x(n)=y(n)/u(n,n);
for k=(n-1):-1:1
    x(k)=(y(k)-u(k,(k+1):n)*x((k+1):n))/u(k,k);
end
x

```

**例 12.8.11** 在 MATLAB 命令窗口中求解例 4.3.1。

**解** >> a=[2 1 5;4 1 12; -2 -4 5]

a =

```

     2     1     5
     4     1    12
    -2    -4     5

```

>> b=[11;27;12]

b =

```

    11
    27
    12

```

>> agui\_lu(a,b)

l =

```

     1     0     0
     2     1     0
    -1     3     1

```

u =

```

     2     1     5
     0    -1     2
     0     0     4

```

y =

```

    11
     5
     8

```

```
x =
     1
    -1
     2
```

计算结果与例 4.3.1 相同。

### 3. 与解线性方程组有关的 MATLAB 数值计算函数

#### (1) 直接矩阵除法求解

**例 12.8.12** 用直接矩阵除法求解例 4.3.1 的线性方程组。

$$\begin{pmatrix} 2 & 1 & 5 \\ 4 & 1 & 12 \\ -2 & -4 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 11 \\ 27 \\ 12 \end{pmatrix}$$

**解** 在 MATLAB 命令窗口中求解例 4.3.1:

```
>> a=[2 1 5;4 1 12;-2 -4 5]
a =
     2     1     5
     4     1    12
    -2    -4     5
>> b=[11;27;12]
b =
    11
    27
    12
>> x=a\b
x =
     1
    -1
     2
```

#### (2) 选列主元 LU 分解函数

格式说明如下:

```
[L,U,P]=lu(a)
```

$L$  为对角线元素全为 1 的下三角矩阵,  $U$  为上三角矩阵,  $P$  为行置换矩阵,  $PA=LU$ 。

**例 12.8.13** 对例 4.3.1 中线性方程组的系数矩阵进行 LU 分解。

**解** >> a=[2 1 5;4 1 12;-2 -4 5]

```
a =
     2     1     5
     4     1    12
    -2    -4     5
>> [l,u,p]=lu(a)
l =
    1.0000         0         0
   -0.5000    1.0000         0
    0.5000   -0.1429    1.0000
u =
    4.0000    1.0000   12.0000
         0   -3.5000    11.0000
         0         0     0.5714
```

```
p =
    0     1     0
    0     0     1
    1     0     0
```

验证:

```
>> inv(p)*l*u
ans =
    2     1     5
    4     1    12
   -2    -4     5
```

### 12.8.3 解线性方程组的迭代法

#### 1. 雅可比迭代法

首先编写 MATLAB 函数文件 agui\_jacobi.m:

```
function x=agui_jacobi(a,b)
% a 为系数矩阵, b 为右端向量, x0 为初始向量(默认为零向量)
% e 为精度(默认为 1e-6), N 为最大迭代次数(默认为 100), x 为返回解向量
n=length(b);
N=100;
e=1e-6;
x0=zeros(n,1);
x=x0;
x0=x+2*e;
k=0;
d=diag(diag(a));
l=-tril(a, -1);
u=-triu(a,1);
while norm(x0-x,inf)>e&&k<N
    k=k+1;
    x0=x;
    x=inv(d)*(l+u)*x+inv(d)*b;
    k
    disp(x')
end
if k==N warning('已达最大迭代次数'); end
```

例 12.8.14 用雅可比迭代法求解例 5.2.1 的线性方程组。

$$\begin{cases} 10x_1 - x_2 - 2x_3 = 72 \\ -x_1 + 10x_2 - 2x_3 = 83 \\ -x_1 - x_2 + 5x_3 = 42 \end{cases}$$

解 在 MATLAB 命令窗口中求解例 5.2.1:

```
>> a=[10 -1 -2; -1 10 -2; -1 -1 5]
a =
    10    -1    -2
    -1    10    -2
    -1    -1     5
```



```
>> b=[72;83;42]
b =
    72
    83
    42
>> x=agui_jacobi(z,b)
计算结果为:
k =     1
    7.200000000000000    8.300000000000000    8.400000000000000
k =     2
    9.710000000000000   10.700000000000000   11.500000000000000
...
k =    16
   10.99999968449670   11.99999968449670   12.99999962583317
x =
   10.99999968449670
   11.99999968449670
   12.99999962583317
```

## 2. 高斯-塞德尔迭代法

首先编写 MATLAB 函数文件 agui\_GS.m:

```
function x=agui_GS(a,b)
% a 为系数矩阵, b 为右端向量, x0 为初始向量(默认为零向量)
% e 为精度(默认为 1e-6), N 为最大迭代次数(默认为 100), x 为返回解向量
n=length(b);
N=100;
e=1e-6;
x0=zeros(n,1);
x=x0;
x0=x+2*e;
k=0;
a1=tril(a);
a2=inv(a1);
while norm(x0-x,inf)>e&k<N
    k=k+1;
    x0=x;
    x=-a2*(a-a1)*x0+a2*b;
    format long
    k
    disp(x')
end
if k==N warning('已达最大迭代次数'); end
```

**例 12.8.15** 在 MATLAB 命令窗口中求解例 5.2.1。

**解** >> a=[10 -1 -2; -1 10 -2; -1 -1 5]

```
a =
    10    -1    -2
    -1    10    -2
    -1    -1     5
```

```
>> b=[72;83;42]
b =
    72
    83
    42
>> x=agui_GS(a,b)
```

计算结果为:

```
k =      1
    7.200000000000000    9.020000000000000    11.644000000000000
k =      2
    10.430800000000000    11.671880000000000    12.820536000000000
...
k =     10
    10.99999996545653    11.99999997883050    12.99999998885741
x =
    10.99999996545653
    11.99999997883050
    12.99999998885741
```

**例 12.8.16** 用 C 语言实现高斯-塞德尔迭代法求解例 5.2.1。

**解** 首先编写 C 语言实现高斯-塞德尔迭代法的程序:

```
#include <math.h>
#include<stdio.h>
#define n 3
#define nmax 100
static double a[n][n]={ {10, -1, -2},{-1,10, -2},{-1, -1,5}};
static double b[n]={72,83,42};
main () {
    int i,j,k;
    double sum,norm,d,s,x[n];
    for(i=0;i<n;i++) x[i]=0;
    k=0;
    printf("\nk=%2dx=",k);
    for(i=0;i<n;i++) printf("%12.8f",x[i]);
    do{
        k++;
        if(k>nmax){printf("\n the iteration failed!");break;}
        norm=0.0;
        for(i=0;i<n;i++){
            s=x[i];
            sum=0.0;
            for(j=0;j<n;j++){
                if(j!=i) sum=sum+a[i][j]*x[j];
            }
            x[i]=(b[i]-sum)/a[i][i];
            d=fabs(x[i]-s);
            if(norm<d) norm=d;
        }
    }
```

```

    printf("\nk=%2d x=",k);
    for(i=0;i<n;i++)printf("%f",x[i]);
} while (norm>=0.1e-6);
if(norm<0.1e-6){
    printf("\n\n the result is:\n");
    printf("\nk=%d",k);
    for(i=0;i<n;i++) printf("x[%d]=%12.8f",i,x[i]);
}
}

```

计算结果为:

k= 0	x= 0.00000000	0.00000000	0.00000000
k= 1	x=7.200000	9.020000	11.644000
k= 2	x=10.430800	11.671880	12.820536
k= 3	x=10.931295	11.957237	12.977706
...			
k=11	x=11.000000	12.000000	13.000000

the result is:

k=11	x[0]= 11.00000000	x[1]= 12.00000000	x[2]= 13.00000000
------	-------------------	-------------------	-------------------

从计算结果可以看出,在精度均为 0.0000001 的前提下,用 C 语言实现高斯-塞德尔迭代法需要 11 次迭代,用 MATLAB 语言实现高斯-塞德尔迭代法需要迭代 10 次。这说明 MATLAB 的计算精度要比 C 语言的计算精度略高。另外,在进行精度控制时, MATLAB 可以直接用迭代前后的解向量近似值的差的范数来控制精度,语句为:

```
norm(x0-x,inf)>e
```

其中,  $\text{norm}(x, \text{inf})$  为向量  $x$  的无穷大范数。

而用 C 语言无论如何都无法做到这一点,因为 C 语言中没有矩阵或向量范数的函数,实现算法时,只能再设计一个循环,通过求迭代前后解向量的各分量差的绝对值的最大值来控制精度,实现起来非常麻烦。语句中为:

```

do{
...
    norm=0.0;
    for(i=0;i<n;i++)
    {s=x[i];
    ...
    d=fabs(x[i]-s);
    if(norm<d)norm=d;
    }
...
} while (norm>=0.1e-6);

```

这说明用 MATLAB 实现计算方法中的算法,比 C 语言要容易得多,也方便得多。

## 12.8.4 函数插值

### 1. 拉格朗日插值

首先编写 MATLAB 函数文件 agui\_lagrange.m:

```

function f=agui_lagrange(x0,y0,x)
%x0 为节点向量, y0 为节点上的函数值, x 为插值点, f 返回插值
n=length(x0);m=length(x);
format long
s=0.0;
for k=1:n
    p=1.0;
    for j=1:n
        if j~=k
            p=p*(x-x0(j))/(x0(k)-x0(j));
        end
    end
    s=p*y0(k)+s;
end
f=s;
end

```

**例 12.8.17** 在 MATLAB 命令窗口中求解例 6.2.2。

**解** x0=[100.0 121.0 144.0]  
x0 =  
100 121 144  
>> y0=[10.0 11.0 12.0]  
y0 =  
10 11 12  
>> f=agui\_lagrange(x0,y0,115.0)  
f =  
10.72275550536420

精确解大约为:

```

>> sqrt(115)
ans =
10.72380529476361

```

**例 12.8.18** 用 C 语言实现拉格朗日插值求解例 6.2.2。

**解** 首先编写实现拉格朗日插值的 C 语言程序:

```

#define N 3
main(){
    float x[N]={100,121,144},y[N]={10,11,12};
    float m,s,a;
    int i,j;
    m=115;
    s=0;
    for(i=0;i<N;i++) {
        a=1;
        for(j=0;j<N;j++) if(i!=j) a=a*(m-x[j])/(x[i]-x[j]);
        s=s+a*y[i];
    }
    printf("\n");
    printf("the result is:\n%15.11f\n",s);
}

```

计算结果为：

the result is:

10.72275543213

Press any key to continue

从计算结果可以看出，实现同样的计算步骤，用 MATLAB 编程的计算结果和 C 语言相比略靠近于精确值，这再一次说明实现同样的算法，MATLAB 的计算精度略高于 C 语言。

## 2. 与插值有关的 MATLAB 函数

### (1) 分段线性插值函数 interp1

格式说明如下：

$y_i = \text{interp1}(x, y, x_i)$

对节点  $(x, y)$  插值，求插值点的函数值。 $x$  为节点向量值， $y$  为对应的节点函数值。

**例 12.8.19** 在 MATLAB 命令窗口中求解例 6.5.1。

**解**  $x = [-5:1:5];$   
 $y = 1./(1+x.^2);$   
 $x_0 = [-5:0.1:5];$   
 $y_0 = \text{agui\_lagrange}(x, y, x_0);$   
 $y_1 = 1./(1+x_0.^2);$   
 $y_2 = \text{interp1}(x, y, x_0);$   
 $\text{plot}(x_0, y_0, '-r')$   
 $\text{hold on}$   
 $\text{plot}(x_0, y_1, '-b')$   
 $\text{hold on}$   
 $\text{plot}(x_0, y_2, '*m')$

结果如图 12.8.1 所示。

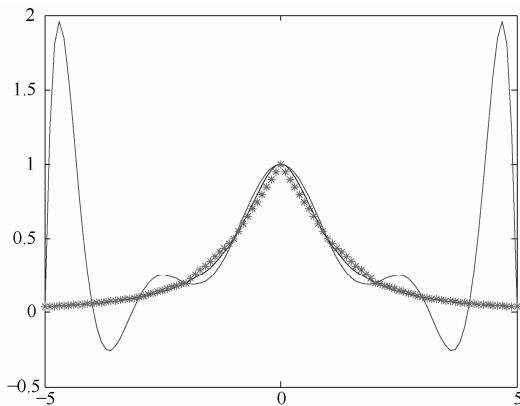


图 12.8.1 例 12.8.19 的结果

### (2) 三次样条插值函数

计算三次样条插值可以用以下函数和命令：

$\text{spline}(x, y, x_i)$  计算插值节点  $x_i$  的三次样条插值。

$\text{spline}(x, y)$  输出一个向量，给出三次样条插值的分段表达式的系数。

$\text{pp} = \text{spline}(x, y); y = \text{ppval}(\text{pp}, x_i)$  计算插值节点  $x_i$  的三次样条插值。

$\text{pp} = \text{spline}(x, y); [\text{node} \text{coef}] = \text{unmkpp}(\text{pp})$  给出由插值节点构成的向量  $\text{node}$ ，以及各段表达式

系数构成的向量 `coef`。

**例 12.8.20** 给定样点(1,1),(2,3),(4,2),(5,2)，用 `spline` 函数求  $f(x)$  的三次样条插值函数，并计算  $f(3)$  的值。

**解** 在 MATLAB 命令窗口中输入：

```
x=[1 2 4 5]
>> y=[1 3 2 2];
>> xi=3;
>> pp=spline(x,y);
>> [nodes codes]=unmkpp(pp)
nodes =
     1     2     4     5
codes =
    0.2500   -1.8333    3.5833    1.0000
    0.2500   -1.0833    0.6667    3.0000
    0.2500    0.4167   -0.6667    2.0000
>> yi=ppval(pp,xi)
yi =
    2.8333
```

由计算结果可知： $f(3)=2.8333$ ，三次样条插值函数的分段表达式为：

$$\begin{cases} s_1 = 0.25(x-1)^3 - 1.8333(x-1)^2 + 3.5833(x-1) + 1 & (1 \leq x \leq 2) \\ s_2 = 0.25(x-2)^3 - 1.0833(x-2)^2 + 0.6667(x-2) + 3 & (2 \leq x \leq 4) \\ s_3 = 0.25(x-4)^3 + 0.4167(x-4)^2 - 0.6667(x-4) + 2 & (4 \leq x \leq 5) \end{cases}$$

用下列指令可画出图形，如图 12.8.2 所示。

```
t=x(1):0.1:x(length(x));
yt=ppval(pp,t);plot(x,y,'ok',t,yt,'-b',xi,yi,'r+')
```

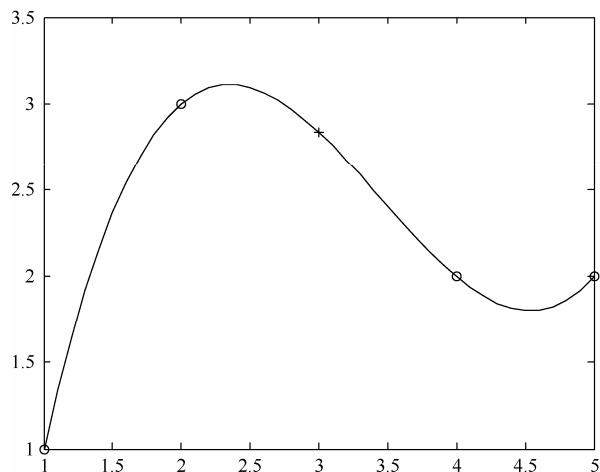


图 12.8.2 例 12.8.20 的图形

## 12.8.5 函数逼近

### 1. 多项式拟合

首先编写 MATLAB 函数文件 `agui_fit.m`：

```

function p=agui_fit(x,y,m)
%x,y 为数据向量, m 为多项式的次数, p 返回多项式的升幂排列系数
A=zeros(m+1,m+1);
for i=0:m
    for j=0:m
        A(i+1,j+1)=sum(x.^(i+j));
    end
    b(i+1)=sum(x.^i.*y);
end
c=A\b';
p=c';

```

**例 12.8.21** 在 MATLAB 命令窗口中求解例 7.5.1。

**解** >> x=[2 4 6 8]  
x =  
 2 4 6 8  
>> y=[2 11 28 40]  
y =  
 2 11 28 40  
>> p=agui\_fit(x,y,1)  
p =  
-12.500000000000000 6.550000000000000

**例 12.8.22** 在 MATLAB 命令窗口中求解例 7.5.2。

**解** >> format rat  
>> x=[-2 -1 0 1 2]  
x =  
 -2 -1 0 1 2  
>> y=[0 1 2 1 0]  
y =  
 0 1 2 1 0  
>> p=agui\_fit(x,y,2)  
p =  
 58/35 0 -3/7

## 2. 与拟合有关的 MATLAB 函数

拟合函数 polyfit 格式说明如下：

```
a=polyfit(x,y,n)
```

x,y 为给定的数值, n 为拟合多项式的次数。

**例 12.8.23** 利用最小二乘法求下列数据的形如  $y = a + bx + cx^2$  的曲线拟合。

x	0.5	1.0	1.5	2.0	2.5	3.0
y	1.75	2.45	3.81	4.80	7.00	8.60

**解** 在 MATLAB 命令窗口中输入：

```

x=[0.5,1.0,1.5,2.0,2.5,3.0];
y=[1.75,2.45,3.81,4.80,7.00,8.60];
a=polyfit(x,y,2)

```

```

a=
0.5614 0.8287 1.1560
x1=[0.5:0.05:3.0];
y1=a(3)+a(2)*x1+a(1)*x1.^2;
plot(x,y,'*')
hold on
plot(x1,y1,'-r')

```

结果如图 12.8.3 所示。

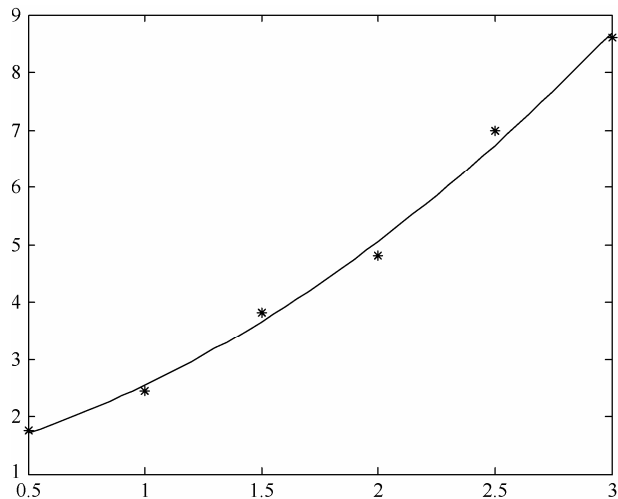


图 12.8.3 例 12.8.23 的结果

## 12.8.6 数值积分

### 1. 复化梯形积分

首先编写 MATLAB 函数文件 `agui_trapz.m` 如下：

```

function t=agui_trapz(fname,a,b,n)
%fname 为被积函数，a,b 分别为下界和上界，n 为等分数
h=(b-a)/n;
fa=feval(fname,a);
fb=feval(fname,b);
f=feval(fname,a+h:h:b-h+0.001*h);
t=h*(0.5*(fa+fb)+sum(f));

```

例 12.8.24 在 MATLAB 命令窗口中求解例 8.3.1。

```

解 >> format long
>> t=agui_trapz(inline('sin(x)./x'),eps,1,8)
t =
0.94569086358270

```

### 2. 复合辛普生求积公式

首先编写 MATLAB 函数文件 `agui_simpson.m`：

```

function s=agui_simpson(fname,a,b,n)
%fname 为被积函数，a,b 分别为下界和上界，n 为等分数

```



```

h=(b-a)/n;
fa=feval(fname,a);
fb=feval(fname,b);
s=fa-fb;
x=a;
for i=1:n
    x=x+h/2;s=s+4*feval(fname,x);
    x=x+h/2;s=s+2*feval(fname,x);
end
s=s*h/6;

```

例 12.8.25 在 MATLAB 命令窗口中求解例 8.3.1。

解 >> t=agui\_simpson(inline('sin(x)./x'),eps,1,4)

```

t =
    0.94608331088847

```

### 3. 龙贝格积分

首先编写 MATLAB 函数文件 agui\_rbg.m:

```

function r=agui_rbg(fname,a,b)
%fname 为被积函数, a,b 分别为下界和上界
e=1e-6;
i=1;j=1;h=b-a;
T(i,1)=h/2*(feval(fname,a)+feval(fname,b));
T(i+1,1)=T(i,1)/2+sum(feval(fname,a+h/2:h:b-h/2+0.001*h))*h/2;
T(i+1,j+1)=4^j*T(i+1,j)/(4^j-1)-T(i,j)/(4^j-1);
while abs(T(i+1,i+1)-T(i,i))>e
    i=i+1;h=h/2;
    T(i+1,1)=T(i,1)/2+sum(feval(fname,a+h/2:h:b-h/2+0.001*h))*h/2;
    for j=1:i
        T(i+1,j+1)=4^j*T(i+1,j)/(4^j-1)-T(i,j)/(4^j-1);
    end
end
T
r=T(i+1,j+1);

```

例 12.8.26 在 MATLAB 命令窗口中求解例 8.4.2。

解 >> agui\_rbg(inline('sin(x)./x'),eps,1)

```

T =
    0.92073549240395         0         0         0
    0.93979328480618    0.94614588227359         0         0
    0.94451352166539    0.94608693395179    0.94608300406367         0
    0.94569086358270    0.94608331088847    0.94608306935092    0.94608307038722
ans =
    0.94608307038722

```

例 12.8.27 用 C 语言实现龙贝格算法求解例 8.4.2。

解 首先编写实现龙贝格算法 C 语言程序:

```

#include <stdio.h>
#include <math.h>

```

```

float f(float m) {
    float y;
    if(m==0) y=1;
    else y=sin(m)/m;
    return (y);
}
void main(){
    float a,b,h,x,S1,S2,T1,T2,s,C1,C2,R1=0,R2=1,eps,d;
    int k=1;
    printf("Please input a,b,eps:");
    scanf("%f,%f,%f",&a,&b,&eps);
    h=b-a;
    T1=h*(f(a)+f(b))/2;
    printf("\nT=%15.10f",T1);
    while(fabs(R2-R1)>eps){
        s=0;
        x=a+h/2;
        while(x<b) { s=s+f(x);x=x+h; }
        T2=T1/2+h*s/2;
        printf("\nT=%15.10f",T2);
        S2=T2+(T2-T1)/3;
        printf("\nS=%15.10f",S2);
        if(k==1) { k=k+1;h=h/2;T1=T2;S1=S2;continue; }
        C2=S2+(S2-S1)/15;
        printf("\nC=%15.10f",C2);
        if(k==2) { C1=C2;k=k+1;h=h/2;T1=T2;S1=S2;continue; }
        R1=d;
        R2=C2+(C2-C1)/63;
        printf("\nR=%15.10f",R2);
        if(k>=3) { d=R2;C1=C2;h=h/2;k=1;T1=T2;S1=S2;continue; }
    }
    printf("\nThe result is:%15.10f\n",R2);
}

```

输入初值:

Please input a,b,eps:0,1,1e-6

计算结果为:

```

T= 0.9207354784
T= 0.9397932887
S= 0.9461458921
T= 0.9445135295
S= 0.9460869829
C= 0.9460830768
T= 0.9456908703
S= 0.9460833073
C= 0.9460830609
R= 0.9460830688
T= 0.9459850192
S= 0.9460830688

```

```

T= 0.9460585415
S= 0.9460830092
C= 0.9460830053
T= 0.9460768700
S= 0.9460829894
C= 0.9460830092
R= 0.9460830092
The result is: 0.9460830092
Press any key to continue

```

在同样的计算精度 0.0000001 下, MATLAB 编程的计算结果为: 0.946 083 070 387 22, C 语言编程的计算结果为: 0.946 083 009 2, 而下面的自适应 Cotes 积分结果为: 0.946 083 070 367 18。这 3 个计算结果中, 小数点之后的 6 位数字都相同, 这说明它们都满足事先给定的计算精度。然而, MATLAB 编程的计算结果与自适应 Cotes 积分结果小数点之后的 10 位数字都相同, 这表明, MATLAB 不但在处理矩阵、向量和数组方面比 C 语言具有很强的优势, 而且其自身的计算精度也非常高。

#### 4. 高斯积分

首先编写 MATLAB 函数文件 agui\_gsjf.m:

```

function g=agui_gsjf(fname,a,b,m)
%%fname 为被积函数, a,b 分别为下界和上界, n 为等分数, m 为高斯点数
switch m
    case 1
        t=0,A=2;
    case 2
        t=[-1/sqrt(3),1/sqrt(3)];A=[1,1];
    case 3
        t=[-sqrt(0.6),0,sqrt(0.6)];A=[5/9,8/9,5/9];
    otherwise
        error('Gauss 点 m 只能取 1,2,3');
end
g=0;
g=g+(b-a)/2*sum(A.*feval(fname,(b-a)/2*t+(a+b)/2));

```

**例 12.8.28** 在 MATLAB 命令窗口中求解例 8.5.1。

**解** >> g=agui\_gsjf(inline('sin(x)./x'),eps,1,3)  
g =  
0.94608313407847

#### 5. 与数值积分有关的 MATLAB 函数

##### (1) 梯形积分

格式说明如下:

```
T = trapz(Y)
```

用等距梯形法近似计算 Y 的积分。若 Y 是一个向量, 则 trapz(Y)为 Y 的积分; 若 Y 是一个矩阵, 则 trapz(Y)为 Y 的每列的积分。

```
T = trapz(X,Y)
```

用梯形法计算  $Y$  在  $X$  点上的积分。若  $X$  为一列向量,  $Y$  为矩阵, 且  $\text{size}(Y,1) = \text{length}(X)$ , 则  $\text{trapz}(X,Y)$  通过  $Y$  的第一个非单元集方向进行计算。

**例 12.8.29** 在 MATLAB 命令窗口中求解例 8.3.1。

```
解 >> x=linspace(eps,1,9);
    >> y=sin(x)./x;
    >> t=trapz(x,y)
    t =
        0.94569086358270
```

(2) 自适应 Simpleson 积分和自适应 Cotes 积分  
格式说明如下。

自适应 Simpleson 积分:

```
q = quad(fun,a,b)
```

近似地从  $a$  到  $b$  计算函数  $\text{fun}$  的数值积分, 误差为  $10^{-6}$ 。若给  $\text{fun}$  输入向量  $x$ , 则应返回向量  $y$ , 即  $\text{fun}$  是一个单值函数。

自适应 Cotes 积分:

```
q=quadl(fun,a,b,tol)
```

与  $\text{quad}$  类似, 但精度高。

**例 12.8.30** 在 MATLAB 命令窗口中求解例 8.3.1。

```
解 >> f=inline('sin(x)./x');
    > q=quad(f,eps,1)
    q =
        0.94608307007653
    >> q=quadl(f,eps,1)
    q =
        0.94608307036718
```

## 12.8.7 常微分方程的数值解法

### 1. 欧拉方法

首先编写 MATLAB 函数文件 `agui_euler.m`:

```
function [x,y]=agui_euler(dfun,span,y0,h)
%dfun 为右端函数,span 为求解区间,y0 为初值,h 为步长,x 返回节点,y 返回数值解
x=span(1):h:span(2);
y(1)=y0;
for n=1:length(x)-1
    y(n+1)=y(n)+h*feval(dfun,x(n),y(n));
end
x=x';y=y';
```

**例 12.8.31** 在 MATLAB 命令窗口中求解例 9.4.4。

```
解 >> dfun=inline('x-y')
    dfun =
        Inline function:
        dfun(x,y) = x-y
    >> [x,y]=agui_euler(dfun,[0,1],0,0.1)
```

计算结果为：

x	y
0	0
0.1	0
0.2	0.010000000000000
0.3	0.029000000000000
0.4	0.056100000000000
0.5	0.090490000000000
0.6	0.131441000000000
0.7	0.178296900000000
0.8	0.230467210000000
0.9	0.287420489000000
1.0	0.34867844010000

## 2. 改进的欧拉方法

首先编写 MATLAB 函数文件 `agui_euler1.m`：

```
function [x,y]=agui_euler1(dfun,span,y0,h)
%dfun 为右端函数,span 为求解区间,y0 为初值,h 为步长,x 返回节点,y 返回数值解
x=span(1):h:span(2);
y(1)=y0;
for n=1:length(x)-1
    k1=feval(dfun,x(n),y(n));
    y(n+1)=y(n)+h*k1;
    k2=feval(dfun,x(n+1),y(n+1));
    y(n+1)=y(n)+h*(k1+k2)/2;
end
x=x';y=y';
```

**例 12.8.32** 在 MATLAB 命令窗口中求解例 9.4.4。

**解** `>> [x,y]=agui_euler1(dfun,[0,1],0,0.1)`

计算结果为：

x	y
0	0
0.1	0.005000000000000
0.2	0.019025000000000
0.3	0.041217625000000
0.4	0.07080195062500
0.5	0.10707576531563
0.6	0.14940356761064
0.7	0.19721022868763
0.8	0.24997525696230
0.9	0.30722760755089
1.0	0.36854098483355

### 3. 四阶龙格-库塔方法

首先编写 MATLAB 函数文件 agui\_RK.m:

```
function [x,y]=agui_RK(dfun,span,y0,h)
%dfun 为右端函数,span 为求解区间,y0 为初值,h 为步长,x 返回节点,y 返回数值解
x=span(1):h:span(2);
y(1)=y0;
for n=1:length(x)-1
    k1=feval(dfun,x(n),y(n));
    k2=feval(dfun,x(n)+h/2,y(n)+h/2*k1);
    k3=feval(dfun,x(n)+h/2,y(n)+h/2*k2);
    k4=feval(dfun,x(n+1),y(n)+h*k3);
    y(n+1)=y(n)+h*(k1+2*k2+2*k3+k4)/6;
end
x=x';y=y';
```

例 12.8.33 在 MATLAB 命令窗口中求解例 9.4.4。

解 >> [x,y]=agui\_RK(dfun,[0,1],0,0.1)

计算结果为:

x	y
0	0
0.1	0.00483750000000
0.2	0.01873090140625
0.3	0.04081842200118
0.4	0.07032028891749
0.5	0.10653093442338
0.6	0.14881193437632
0.7	0.19658561867123
0.8	0.24932928973443
0.9	0.30656999120008
1.0	0.36787977441250

例 12.8.34 用 C 语言实现四阶龙格-库塔方法求解例 9.4.4。

解 首先编写实现四阶龙格-库塔方法的 C 语言程序:

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
main () {
    float x0,y0,h,b;
    double k1,k2,k3,k4,x1,y1;
    int i=0,n;
    double f(double x,double y);
    printf("\nPlease input x0,y0,h,b:");
    scanf("%f,%f,%f,%f",&x0,&y0,&h,&b);
    n=(int)((b-x0)/h)+1;
    for (i=1;i<=n;i++){
```

```

        x1=x0+h;
        k1=f(x0,y0);
        k2=f(x0+h/2.0,y0+h*k1/2.0);
        k3=f(x0+h/2.0,y0+h*k2/2.0);
        k4=f(x1,y0+h*k3);
        y1=y0+h*(k1+k2*2.0+k3*2.0+k4)/6.0;
        y0=y1;
        x0=x1;
        printf("\n%f,%f\n",x0,y0);
    }
}
double f(double x,double y){
    double z;
    z=x-y;
    return(z);
}

```

输入初值:

Please input x0,y0,h,b:0,0,0.1,1

计算结果为:

```

0.10000    0.00483750
0.20000    0.01873090
0.30000    0.04081842
0.40000    0.07032029
0.50000    0.10653094
0.60000    0.14881194
0.70000    0.19658563
0.80000    0.24932930
0.90000    0.30656999
1.00000    0.36787978

```

Press any key to continue

用两种语言编程的计算结果基本相同，但 MATLAB 表示结果的位数要多。

#### 4. 与求解常微分方程初值问题有关的 MATLAB 函数

求解常微分方程初值问题的龙格-库塔方法的函数为 ode23 和 ode45。

格式如下:

```

[t,y]=ode23('f',tspan,y0)
[t,y]=ode45('f',tspan,y0)

```

其中，f 为方程右端函数的函数名，tspan 为求解区间，y0 为解函数的初值，输出结果 t 为离散节点  $a = x_0 < x_1 < \dots < x_n = b$ ，y 为相应离散节点上的数值解。

**例 12.8.35** 求例 9.2.2 的一阶常微分方程初值问题的解。

$$\begin{cases} \frac{dy}{dx} = 2xy & 0 \leq x \leq 1 \\ y(0) = 1 \end{cases}$$

**解** 首先编写 MATLAB 函数文件:

```

function f=fun(t,y)
f=2*t*y

```

在 MATLAB 命令窗口中输入:

```
t=0:0.1:1;
[t,y]=ode23('fun',t,1);
[t,y]
ans =
      0      1.0000
    0.1000    1.0100
    0.2000    1.0408
    0.3000    1.0942
    0.4000    1.1735
    0.5000    1.2840
    0.6000    1.4333
    0.7000    1.6322
    0.8000    1.8963
    0.9000    2.2476
    1.0000    2.7177
```

```
[t,y]=ode45('fun',t,1)
[t,y]
ans =
      0      1.0000
    0.1000    1.0100
    0.2000    1.0408
    0.3000    1.0942
    0.4000    1.1735
    0.5000    1.2840
    0.6000    1.4333
    0.7000    1.6322
    0.8000    1.8963
    0.9000    2.2476
    1.0000    2.7177
```

与精确解  $y = e^{x^2}$  的值相比, 比改进的欧拉方法精度高。

## 12.8.8 矩阵特征值问题计算

### 1. 幂法

首先编写 MATLAB 函数文件 aguimifa:

```
function [tzz,v,k]=aguimifa(A,x0,eps,max0)
%A 为 n 阶方阵,x0 为初值,eps 为上限,max0 为循环次数
%tzz 为主特征值,v 为特征向量,k 为迭代次数
tzz=0;k=0;err=1;
while ((k<max0)&(err>eps))
    y=A*x0
    [m,j]=max(abs(y))
    m=y(j)
    x0=y/m
```



```

        err=abs(tzz-m)
        tzz=m
        k=k+1
    end
    v=x0;
end

```

**例 12.8.36** 用幂法求例 10.2.1 矩阵  $A = \begin{pmatrix} 3 & 2 \\ 4 & 5 \end{pmatrix}$  的主特征值和对应的特征向量。

**解** 在 MATLAB 命令窗口中输入：

```

A=[3,2;4,5];
x=[1;1];
[tzz,v,k]=aguimifa(A,x,1e-10,100)
tzz =
    7.0000000000015481
v =
    0.5000000000000553
    1.0000000000000000
k =
    14

```

## 2. 带原点位移的反幂法

首先编写 MATLAB 函数文件 aguifanmifa:

```

function [tzz,v,k]=aguifanmifa(A,x,p,esp,max0)
%A 为 n 阶矩阵,x 为初始向量,p 为可选参数,esp 为精度,max0 为最大迭代次数
%tzz 为返回的特征值,v 为返回的特征向量,k 为迭代次数
tzz=0;k=0;err=1;
mu=0.5;n=length(x);
A=A-p*eye(n);
[L,U,P]=lu(A);
while ((k<max0)&&(err>esp))
    [m,j]=max(abs(x));
    m=x(j);y=x./m;
    z=L\((P*y);x=U\z;
    err=abs(m-mu);
    k=k+1;mu=m;
end
tzz=p+1/m;
v=y;
end

```

**例 12.8.37** 用带原点位移的反幂法求例 10.2.2。

在 MATLAB 命令窗口中输入：

```

A=[-12,3,3;3,1,-2;3,-2,7]
A =
   -12     3     3
     3     1    -2
     3    -2     7

```

```

p=-13
p =
    -13
[tzz,v,k]=aguifanmifa(A,x,p,1e-10,100)
tzz =
    -13.220179976292638
v =
    1.0000000000000000
   -0.235105487306686
   -0.171621171457527
k =
    12

```

### 3. QR 方法

首先编写 MATLAB 函数文件 aguihouse, aguiqr, aguiqrmaxtr:

```

function [H,rho]= aguihouse (x,y)
% 求解正交对称的 Householder 矩阵 H,使 Hx=rho*y,其中 rho=-sign(x(1))*||x||/||y||
% 参数说明:x 为列向量,y 为列向量,x 和 y 必须具有相同的维数
t=-sign(x(1));
if (t==0) t=1;end
rho=t*norm(x)/norm(y);
y=rho*y;
v=(x-y)/norm(x-y);
I=eye(length(x));
H=I-2*v*v';

function [Q,R]=aguiqr(A)
% 基于 Householder 变换, 将方阵 A 分解为 A=QR, 其中 Q 为正交矩阵, R 为上三角阵
n=size(A,1);
R=A;
Q=eye(n);
for i=1:n-1
    x=R(i:n,i);
    y=[1;zeros(n-i,1)];
    Ht=aguihouse (x,y);
    H=blkdiag(eye(i-1),Ht);
    Q=Q*H;
    R=H*R;
end
function [k T]= aguiqrmaxtr (A,tol)
%求 A 的全部特征值, T 为特征值, k 为迭代次数
%本程序只能求实特征值, 求虚特征值需要进一步处理
k=1;
n=length(A);
A0=zeros (n);
while norm (diag(A-A0))>tol
    k=k+1;

```

```

A0=A;
[Q,R]=aguiqr (A);
A=R*Q;
end
k
T=diag(A)

```

例 12.8.38 用反射变换, 将例 10.3.2 中的矩阵  $A = \begin{pmatrix} 0 & 2 & 0 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{pmatrix}$  进行 QR 分解。

解 在 MATLAB 命令窗口中输入:

```

>> A=[0,2,0;2,1,2;0,2,1]
A =
     0     2     0
     2     1     2
     0     2     1
>> [q,r]=aguiqr (A)

q =
    0.0000   -0.7071   -0.7071
    1.0000   -0.0000   -0.0000
         0   -0.7071    0.7071
r =
    2.0000    1.0000    2.0000
   -0.0000   -2.8284   -0.7071
   -0.0000    0.0000    0.7071

```

在 MATLAB 中, 也可以直接调用程序进行 QR 分解, 在命令窗口中输入:

```

>> A=[0,2,0;2,1,2;0,2,1]
A =
     0     2     0
     2     1     2
     0     2     1
>> [q,r]=qr (A)
q =
     0    0.7071   -0.7071
    1.0000         0         0
     0    0.7071    0.7071
r =
    2.0000    1.0000    2.0000
     0    2.8284    0.7071
     0         0    0.7071

```

例 12.8.39 用 QR 方法求例 10.3.3 中矩阵  $A = \begin{pmatrix} 0 & 2 & 0 \\ 2 & 1 & 2 \\ 0 & 2 & 1 \end{pmatrix}$  的特征值。

解 在 MATLAB 命令窗口中输入:

```
>> A=[0,2,0;2,1,2;0,2,1]
A =
     0     2     0
     2     1     2
     0     2     1
>> [k T]= aguiqrmxtr (A,1e-5)
k =
    16
T =
    3.6262
   -2.1413
    0.5151
```

#### 4. 雅可比方法

首先编写 MATLAB 函数文件：

```
function [k,d,v]=aguiykb(a,tol)
% Jacobi 方法求特征值
% 输入：A 为方阵
% 输出：k 为迭代次数，d 为特征值，v 为特征向量
[m,n]=size(a);
a1=a;
k=1;
b=a1-diag(diag(a1));
[m1,i]=max(abs(b));
[m2,j]=max(m1);
i=i(j);
v=eye(m,n);
while sum(sum(b.*b))>tol
    k=k+1;
    x=(a1(i,i)-a1(j,j))/(2.0*a1(i,j));
    u=acot(x)/2;
    v1=eye(m,n);
    v1(i,i)=cos(u);
    v1(j,j)=cos(u);
    v1(i,j)=-sin(u);
    v1(j,i)=sin(u);
    v=v*v1;
    a1=v1'*a1*v1
    b=a1-diag(diag(a1));
    [m1,i]=max(abs(b));
    [m2,j]=max(m1);
    i=i(j);
end
v=v';
d=diag(a1)
k
```

**例 12.8.40** 用雅可比方法计算例 10.4.1 中对称矩阵  $A$  的特征值和对应的一组特征向量。

$$A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$$

在 MATLAB 命令窗口中输入：

```
>> A=[2, -1,0; -1,2, -1;0, -1,2]
A =
     2     -1     0
    -1     2    -1
     0     -1     2
>> [k,d,v]=aguiykb(A,1e-5)
v =
     0.5004     0.7066     0.5004
    -0.4997     0.7076    -0.4996
    -0.7071    -0.0000     0.7071
d =
     0.5858
     3.4142
     2.0000
k =
     7
```

## 5. 与求解特征值问题有关的 MATLAB 函数

只求解特征值或者求解特征值与特征向量的函数 **eig** 格式说明如下：

$d = \text{eig}(A)$  或  $[v,d] = \text{eig}(A)$

$d$  为特征值， $v$  为特征向量。

**例 12.8.41** 求例 10.4.1 中对称矩阵  $A$  的特征值和对应的一组特征向量。

**解** >> A=[2, -1,0; -1,2, -1;0, -1,2]

```
A =
     2     -1     0
    -1     2    -1
     0     -1     2
>> d=eig(A)
d =
     0.585786437626905
     2.000000000000000
     3.414213562373095
>> [v,d]=eig(A)
v =
     0.500000000000000    -0.707106781186547    -0.500000000000000
     0.707106781186547     0.000000000000000     0.707106781186547
     0.500000000000000     0.707106781186547    -0.500000000000000
d =
     0.585786437626905         0         0
         0     2.000000000000000         0
         0         0     3.414213562373095
```

## 12.8.9 函数优化计算

### 1. 梯度法

例 12.8.42 用梯度法求解例 11.3.4。

解 首先编写 MATLAB 函数文件 aguitidu:

```
function f = aguitidu(x,esp)
%x 为初值, esp 为精度
k=0;
while(sqrt((1.6*x(1)).^2+2*x(2).^2)>esp)
    t=(3.2*x(1).^2+4*x(2).^2)/(2*1.6^2*x(1).^2+8*x(2).^2)
    x(1)=x(1)-t*1.6*x(1);
    x(2)=x(2)-t*2*x(2);
    k=k+1
x
end
f=0.8*x(1).^2+x(2).^2
```

在 MATLAB 命令窗口中输入:

```
>> x=[1,1]
x =
    1    1
>> esp=0.05
esp = 0.0500000000000000
>> f = aguitidu(x,esp)
t =
    0.548780487804878
k = 1
x =    0.121951219512195   -0.097560975609756
t =    0.562500000000000
k = 2
x =    0.012195121951219    0.012195121951220
f =    2.676977989292087e-004
```

### 2. 牛顿法

例 12.8.43 用牛顿法求解例 11.3.6。

解 首先编写 MATLAB 函数文件 aguiniudun:

```
function f = aguiniudun(x,esp)
%x 为初值, esp 为精度
k=1;
while(sqrt((3*x(1)*x(1)-1).^2+(3*x(2)*x(2)-1).^2)>esp)
    x0=[3*(x(1)*x(1)-1);3*(x(2)*x(2)-1)];
    x1=[6*x(1),0;0,6*x(2)];
    x=x-inv(x1)*x0
    k=k+1
end
f=x(1).^3+x(2).^3-3*(x(1)+x(2))
```

在 MATLAB 命令窗口中输入:

```
>> x=[6;4]
x =
     6
     4
>> esp=0.05
esp = 0.0500000000000000

>> f = aguiniudun(x,esp)
x =
    3.083333333333334
    2.125000000000000
k = 1
x =
    1.703828828828829
    1.297794117647059
k = 2
x =
    1.145371122940521
    1.034166180636561
k = 3
x =
    1.009225290808246
    1.000564381199631
k = 4
x =
    1.000042164019903
    1.000000159173235
k = 5
x =
    1.000000000888865
    1.000000000000013
k = 6
x =
     1
     1
k = 7
f = -4
```

### 3. 共轭方向法

例 12.8.44 用共轭方向法解例 11.3.7。

解 首先编写 MATLAB 函数文件 aguigongce:

```
function f=aguigongce(x,esp)
%x 为初值, esp 为精度
g=[2*x(1),50*x(2)];
s=-g;
k=0;
while(sqrt((2*x(1)).^2+(50*x(2)).^2)>esp)
t=(4*x(1).^2+2500*x(2).^2)/(8*x(1).^2+125000*x(2).^2)
```

```

x0=x;
x(1)=x(1)+t*s(1);
x(2)=x(2)+t*s(2);
I=((2*x(1)).^2+(50*x(2)).^2)/((2*x0(1)).^2+(50*x0(2)).^2);
g=[2*x(1),50*x(2)];
s=-g+I.*s;
k=k+1
x
end
f=x(1).^2+25*x(2).^2;

```

在 MATLAB 命令窗口中输入：

```

>> x=[1,1]
x =      1      1
>> esp=0.05
esp =    0.0500000000000000
>> f=aguigong(x,esp)
t =    0.020030718034046
k =      1
x =    0.959938563931908   -0.001535901702291
t =    0.481538461538460
k =      2
x =    0.034023952487030   -0.000054438323978
t =    0.481538461539468
k =      3
x =    0.000093071178028    0.001258130131412
t =    0.020000168112634
k =      4
x =   -0.001113106914784    0.000046504714604
f =    1.293074215748666e-006

```

#### 4. 与函数优化计算有关的 MATLAB 函数

求解无约束最优化函数 fminsearch 格式说明如下：

```
[x,y]=fminsearch(@fun,x0)
```

x 为最优解，y 为函数值，x0 为初值。

**例 12.8.45** 求例 11.2.1 中的问题： $\min f(x) = x^5 + x^3 - 7x$ 。

**解** function aguifmin()

```
x0=1;[x,y]=fminsearch(@fun,x0)
```

```
function y=fun(x)
```

```
y=x^5+x^3-7*x;
```

```
>> aguifmin
```

```
x = 0.959472656250000
```

```
y = -5.019894124369483
```

## 本章小结

本章简要介绍现代数值计算软件 MATLAB，包括 MATLAB 的编程环境、基本命令、矩阵和向量的数值计算、符号微积分和符号方程的计算、二维和三维图形显示功能、程序的控制流



程和调试方法。与其他语言（如 C 语言）相比，MATLAB 具有明显的矩阵处理和图形处理的优势。本章给出了约 40 个 MATLAB 源代码和 6 个 C 语言程序源代码，实现了本书中的主要算法，测试数据也全部采用本书中的例题。通过比较两种语言的差异，可以帮助学生更好地学习 MATLAB 软件和计算方法课程。

## 习题 12

12.1 数组  $x, y$  定义为  $x=[7 \ 4 \ 3]$ ,  $y=[-1 \ -2 \ -3]$ , 要求:

(1) 将  $y$  加到  $x$  之前产生新的数组  $u$ 。

(2) 将  $y$  加到  $x$  之后产生新的数组  $v$ 。

12.2 给定一个向量:

$$a=[4 \ -1 \ 2 \ -8 \ 4 \ 5 \ -3 \ -1 \ 6 \ -7]$$

编写一个程序, 使  $a$  中的负元素加倍, 运行程序, 并显示结果。

12.3 给定一个向量:

$$a=[4 \ -1 \ 2 \ -8 \ 4 \ 5 \ -3 \ -1 \ 6 \ -7]$$

编写一个程序, 计算  $a$  中正元素的和, 运行程序, 并显示结果。

12.4 给定数组  $x=[1:99]$ , 编写程序移除  $x$  中的所有素数, 然后计算其元素之和。

12.5 编写一个函数 M 文件 `fun_ex(x)`, 计算如下函数:

$$y = 0.8e^{-x} + x^3 \sin x$$

12.6 编写一个函数 M 文件 `fun_es(x)`, 计算如下函数:

$$f(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!}$$

12.7 在指定的定义域内, 绘制下列函数的图形:

$$y = \frac{1}{1+(x-2)^2} \quad (0 \leq x \leq 4)$$

12.8 已知:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \\ 3 & 0 & 2 \end{pmatrix}, \quad B = \begin{pmatrix} 4 & 1 & 2 \\ 3 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$

计算  $C=A+B$ ,  $D=A-B$ ,  $E=AB$ ,  $F=BA$ 。

12.9 已知:

$$B = \begin{pmatrix} 4 & 1 & 2 \\ 3 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$

计算  $B$  的行列式和  $B$  的逆矩阵, 并验证  $BB^{-1}$ 。

12.10 用 `poly` 命令将下列多项式表示为幂级数形式:

$$y=5(x-3)(x-4)(x+1)(x+3)$$

## 附录 A 计算方法实验

计算方法实验是学习计算方法课程的重要环节，通过编程实现一些典型的实验题目，可以加深对所学内容的理解，进一步了解相关算法的特点和适用范围。每次实验完成后，都要以实验报告的形式完成。实验报告除了写明专业、年级、班级、学号和姓名等必要的信息外，还应该包括如下内容。

**实验目的：**写清楚为什么做这个实验，其目的是什么，做完这个实验要达到什么结果，实验的注意事项是什么等。

**实验方法：**写清楚本实验所涉及的算法原理及理论基础等。

**实验内容：**填写实验题目、数据准备、算法流程及实施方案等。

**实验程序：**将实验用到的程序填写完整并添加一定的注释。

**实验结果：**实验结果应包含使用的原始数据、中间结果和最终结果，复杂的结果可以用表格或图形表示，较为简单的结果可以与结果分析合并出现。

**结果分析：**对实验结果进行认真分析，进一步明确实验所涉及的算法的优缺点和使用范围，最后进行必要的误差分析。

下面给出了实验报告的一个样板，仅供参考。

××大学××学院

### 实验报告

姓 名		学 号		专业班级		
课程名称					实验日期	
成 绩		指导老师		批改日期		
实 验 名 称						
一、实验目的						
二、实验方法						
三、实验内容						
四、实验程序						
五、实验结果						
六、结果分析						
教师 评语						

## 实验 1 方程求根

### 一、实验目的

用不同方法求任意实函数方程  $f(x)=0$  在自变量区间  $[a,b]$  内或某一点附近的实根。并比较方法的优劣性。

### 二、实验方法

#### (1) 二分法

对方程  $f(x)=0$  在  $[a,b]$  内求根。将所给区间二分，在分点  $x=\frac{b-a}{2}$  处判断是否  $f(x)=0$ 。若是，则有根  $x=\frac{b-a}{2}$ 。否则继续判断是否  $f(a)\cdot f(x)<0$ 。若是，则令  $b=x$ ，否则令  $a=x$ 。重复此过程，直至求出方程  $f(x)=0$  在  $[a,b]$  内的近似根为止。

#### (2) 迭代法

将方程  $f(x)=0$  等价变换为  $x=\varphi(x)$  形式并建立相应的迭代公式  $x_{k+1}=\varphi(x_k)$ 。

#### (3) 牛顿法

设已知方程  $f(x)=0$  的一个近似根  $x_0$ ，则函数  $f(x)$  在点  $x_0$  附近可用一阶泰勒多项式  $p_1(x)=f(x_0)+f'(x_0)(x-x_0)$  来近似，因此方程  $f(x)=0$  可近似表示为  $f(x_0)+f'(x_0)(x-x_0)=0$ 。

设  $f'(x_0)\neq 0$ ，则  $x=x_0-\frac{f(x_0)}{f'(x_0)}$ 。取  $x$  作为原方程新的近似根  $x_1$ ，然后再将  $x_1$  作为  $x_0$  代入上式。

迭代公式为：
$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)}。$$

### 三、实验内容

(1) 在区间  $[0,1]$  内用二分法求方程  $e^x+10x-2=0$  的近似根，要求误差不超过  $0.5\times 10^{-3}$ 。

(2) 取初值  $x_0=0$ ，用迭代公式  $x_{k+1}=\frac{2-e^{x_k}}{10}$  ( $k=0,1,2,\dots$ ) 求方程  $e^x+10x-2=0$  的近似根，要求误差不超过  $0.5\times 10^{-3}$ 。

(3) 取初值  $x_0=0$ ，用牛顿迭代法求方程  $e^x+10x-2=0$  的近似根，要求误差不超过  $0.5\times 10^{-3}$ 。

### 四、实验程序

(略)

### 五、实验结果 (仅供参考)

(1)  $x_{11}=0.09033$     (2)  $x_5=0.09052$     (3)  $x_2=0.09052$

## 六、结果分析

(略)

提示：比较三种方法的计算量。

## 实验 2 解方程组的直接法

### 一、实验目的

用高斯消去法解线性方程组  $Ax = b$ 。式中， $A$  为  $n$  阶非奇异方阵， $x, b$  是  $n$  阶列向量，并分析选主元的重要性。

### 二、实验方法

#### (1) 顺序消去法

通过变换，将系数矩阵转换成等价的上三角矩阵，经过回代求出方程组的解。

#### (2) 列主元消去法

在第  $i$  步时，首先将  $a_{ij} (j < i)$  化为 0。在第  $i$  列余下的  $a_{ij} (j \geq i)$  中选择绝对值最大的元素作为主元，且把它所在的行和第  $i$  行交换，同时将列下标的交换记录下来。其次将  $a_{ki} (k < i)$  化为 0，

然后利用回代公式  $x_i = b_i^{(k)} - \sum_{j=i+1}^n a_{i,j}^{(k)} x_j$  ( $i = n, n-1, \dots, 1$ ) 进行回代，求出方程组的解。

### 三、实验内容

解下列方程组：

$$\begin{pmatrix} 1.1348 & 3.8326 & 1.1651 & 3.4017 \\ 0.5301 & 1.7875 & 2.5330 & 1.5435 \\ 3.4129 & 4.9317 & 8.7643 & 1.3142 \\ 1.2371 & 4.9998 & 10.6721 & 0.0147 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 9.5342 \\ 6.3941 \\ 18.4231 \\ 16.9237 \end{pmatrix}$$

### 四、实验程序

(略)

### 五、实验结果 (仅供参考)

(1)  $(1.0138, 0.99689, 1.0000, 0.99891)^T$

(2)  $(0.99987, 1.0001, 1.0000, 0.9996)^T$

精确解为： $(1, 1, 1, 1)^T$

## 六、结果分析

(略)

提示：选主元对结果的影响。

## 实验 3 解线性方程组的迭代法

### 一、实验目的

用雅可比迭代法和高斯-塞德尔迭代法解线性方程组  $Ax=b$ ，式中  $A$  为非奇异实矩阵。在给定迭代初值的情况下，进行迭代，直到满足精度要求。

### 二、实验方法

(1) 雅可比迭代法

设系数矩阵  $A$  为非奇异矩阵，且  $a_{ii} \neq 0 (i=1,2,\dots,n)$ ，从第  $i$  个方程中解出  $x_i$ ，得其等价形式：

$$x_i = \frac{1}{a_{ii}} \left( b - \sum_{j=1, j \neq i}^n a_{ij} x_j \right)$$

取初始向量  $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)})$ ，可建立相应的迭代公式：

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( - \sum_{j=1, j \neq i}^n a_{ij} x_j^{(k)} + b_i \right)$$

(2) 高斯-塞德尔迭代法

每计算出一个新的分量便立即用它取代对应的旧分量进行迭代，可能收敛更快，据此思想可构造高斯-塞德尔迭代法，其迭代公式为：

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)} + b_i \right) \quad (i=1,2,\dots,n)$$

### 三、实验内容

求下列线性方程组的近似解及相应的迭代次数：

$$\begin{pmatrix} 4 & -1 & 0 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 \\ -1 & 0 & -1 & 4 & -1 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 0 & -1 & 4 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 5 \\ -2 \\ 5 \\ -2 \\ 6 \end{pmatrix}$$

要求：  $\|x^{(k+1)} - x^{(k)}\|_2 \leq 0.0001$ ，初值选为常向量  $b$ 。

## 四、实验程序

(略)

## 五、实验结果

精确解为:  $(1, 2, 1, 2, 1, 2)^T$

## 六、结果分析

(略)

提示: 比较两种方法的迭代次数。

## 实验 4 插值问题

### 一、实验目的

用拉格朗日插值和牛顿插值的方法, 在已知函数在点  $x_0, x_1, \dots, x_n$  处的函数值  $y_0, y_1, \dots, y_n$  的情况下, 求插值节点  $x$  的函数值  $y$ , 即求  $f(x)$ 。比较结果, 并说明为什么相等。

### 二、实验方法

(1) 拉格朗日插值

根据  $x_0, x_1, \dots, x_n; y_0, y_1, \dots, y_n$  构造插值多项式

$$p_n(x) = \sum_{k=0}^n \left( \prod_{\substack{j=0 \\ j \neq k}}^n \frac{x - x_j}{x_k - x_j} \right) \cdot y_k$$

将插值点  $x$  代入上式, 可得函数  $f(x)$  在点  $x$  处函数值的近似值。

(2) 牛顿插值

根据  $x_0, x_1, \dots, x_n; y_0, y_1, \dots, y_n$  构造插值多项式

$$N_n(x) = f(x_0) + f(x_0, x_1)(x - x_0) + \dots + f(x_0, x_1, \dots, x_n)(x - x_0)(x - x_1) \cdots (x - x_{n-1})$$

牛顿插值公式中各项的系数就是函数  $f(x)$  的各阶差商 (均差)

$$f(x_0), f(x_0, x_1), \dots, f(x_0, x_1, \dots, x_n)$$

因此, 在构造牛顿插值公式时, 常常先把差商列成一个表, 称为差商表。

### 三、实验内容

从以下函数表:

$x$	0.4	0.55	0.8	0.9	1
$f(x)$	0.41075	0.57815	0.88811	1.02652	1.17520

出发, 计算  $f(0.5)$ ,  $f(0.7)$  及  $f(0.85)$  的近似值。

## 四、实验程序

(略)

## 五、实验结果 (仅供参考)

$$f(0.5)=0.521090, f(0.7)=0.758589, f(0.85)=0.956119$$

## 六、结果分析

(略)

提示: 比较两种方法的计算量和优劣。

# 实验 5 曲线拟合

## 一、实验目的

用最小二乘法, 在已知函数在点  $x_0, x_1, \dots, x_n$  处的函数值  $y_0, y_1, \dots, y_n$  的情况下, 求拟合多项式。

## 二、实验方法

(1) 由给定的数据在坐标纸上进行描点, 根据点的分布, 确定拟合多项式的次数  $m$ 。

(2) 求解正规方程组:

$$\begin{pmatrix} \sum 1 & \sum x_i & \cdots & \sum x_i^m \\ \sum x_i & \sum x_i^2 & \cdots & \sum x_i^{m-1} \\ \vdots & \vdots & \vdots & \vdots \\ \sum x_i^m & \sum x_i^{m+1} & \cdots & \sum x_i^{2m} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum x_i y_i \\ \vdots \\ \sum x_i^m y_i \end{pmatrix}$$

求出系数  $a_i^*$  ( $i=0, 1, 2, \dots, m$ )。

(3) 写出拟合多项式  $S(x) = \sum_{i=0}^m a_i x^i$ 。

## 三、实验内容

已知一组实验数据:

$i$	1	2	3	4	5	6	7	8	9
$x_i$	1	3	4	5	6	7	8	9	10
$y_i$	10	5	4	2	1	1	2	3	4

试用最小二乘法求它的多项式拟合曲线, 并画出图形。

## 四、实验程序

(略)

## 五、实验结果 (仅供参考)

$$y = 13.4597 - 3.6053x + 0.2676x^2$$

## 六、结果分析

(略)

提示: 比较用正交多项式拟合能否提高精度。

## 实验 6 数值积分

### 一、实验目的

利用复化梯形公式、复化辛普生公式和龙贝格数值积分公式计算  $\int_a^b f(x)dx$  的近似值。

### 二、实验方法

(1) 将  $[a, b]$  区间  $n$  等分, 记分点为  $x_i = a + ih$  ( $h = \frac{b-a}{n}$ ;  $i = 0, 1, \dots, n$ ), 并在每个小区间  $[x_i, x_{i+1}]$  内应用梯形公式

$$T_n = \frac{h}{2} \left[ f(a) + 2 \sum_{i=1}^{n-1} f(x_i) + f(b) \right]$$

(2) 在每个小区间  $[x_i, x_{i+1}]$  内, 应用辛普生公式

$$S_n = \frac{h}{6} \left[ f(a) + 4 \sum_{i=0}^{n-1} f(x_{i+\frac{1}{2}}) + 2 \sum_{i=1}^{n-1} f(x_i) + f(b) \right]$$

式中  $x_{i+\frac{1}{2}}$  为  $[x_i, x_{i+1}]$  的中点, 即  $x_{i+\frac{1}{2}} = x_i + \frac{1}{2}h$ 。

(3) 先用梯形公式计算  $T_1 = (b-a)/2 \times [f(a) + f(b)]$ , 然后, 采用将求积区间  $(a, b)$  逐次折半的办法, 令区间长度  $h = (b-a)/2^i$  ( $i = 0, 1, 2, \dots$ ), 计算

$$T_{2n} = \frac{1}{2}T_n + \frac{h}{2} \sum_{k=1}^n f(a + h(k - \frac{1}{2}))$$

式中  $n = 2^i$ 。

于是, 得到辛普生公式  $S_n = T_{2n} + (T_{2n} - T_n)/3$

得到柯特斯求积公式  $C_n = S_{2n} + (S_{2n} - S_n)/15$

最后, 得到龙贝格求积公式  $R_n = C_{2n} + (C_{2n} - C_n)/63$

利用上述各公式计算, 直到相邻两次的积分结果之差满足精度要求。



### 三、实验内容

利用复化梯形公式、复化辛普生公式和龙贝格数值积分法计算

$e^2 = \int_1^2 xe^x dx$  和  $\pi = \int_0^1 \frac{4}{1+x^2} dx$  的近似值, 要求误差为  $\varepsilon = \frac{1}{2} \times 10^{-7}$ , 将计算结果与精确值比

较, 并对计算结果进行分析 (计算量、误差)。

### 四、实验程序

(略)

### 五、实验结果

(略)

### 六、结果分析

(略)

提示: 比较不同方法的计算量和优劣。

## 实验 7 数值微分

### 一、实验目的

用变步长的中点方法和三点求导公式求  $e^x$  在  $x=1$  处的导数值, 并比较步长的变化对解的影响。

### 二、实验方法

(1) 变步长的中点方法计算公式为

$$G(h) = \frac{e^{1+h} - e^{1-h}}{2h}$$

式中步长  $h = \frac{h}{2^k}$ ,  $k$  为二分次数。

(2) 分别取  $i=0,1,2$ , 带余项的三点求导式为

$$\begin{cases} f'(x_0) = \frac{1}{2h}[-3f(x_0) + 4f(x_1) - f(x_2)] + \frac{h^2}{3}f'''(\xi) \\ f'(x_1) = \frac{1}{2h}[-f(x_0) + f(x_2)] - \frac{h^2}{6}f'''(\xi) \\ f'(x_2) = \frac{1}{2h}[f(x_0) - 4f(x_1) + 3f(x_2)] + \frac{h^2}{3}f'''(\xi) \end{cases} \quad \xi \in [x_0, x_1]$$

求  $h$  分别取 1, 0.1, 0.01 的计算结果。

### 三、实验内容

- (1) 用变步长的中点方法求  $e^x$  在  $x=1$  处的导数值, 步长从  $h=0.8$  开始。  
(2) 用三点求导公式,  $h$  分别取 1, 0.1, 0.01, 计算  $e^x$  在  $x=1$  处的结果。

### 四、实验程序

(略)

### 五、实验结果 (仅供参考)

- (1) 二分 9 次得到的结果  $f'(1) \approx 2.71828$ , 有 6 位有效数字。

- (2)  $h=1$  时, 取  $x_0=1, x_1=1, x_2=2$ ,  $f'(1) \approx \frac{1}{2}(-e^0 + e^2) = 3.195$

$$h=0.1 \text{ 时, 取 } x_0=0.90, x_1=1.00, x_2=1.10, \quad f'(1) \approx \frac{1}{2 \times 0.1}(-e^{0.90} + e^{1.10}) = 2.720$$

$$h=0.001 \text{ 时, 取 } x_0=0.99, x_1=1.00, x_2=1.01, \quad f'(1) = \frac{1}{2 \times 0.01}(-e^{0.99} + e^{1.01}) = 2.750$$

(中间结果取小数点后 3 位有效数字)

$f'(1)$  的真值为  $e^1 = 2.7182818$ 。

### 六、结果分析

(略)

提示: 分析步长  $h$  的选取对计算结果的影响。

## 实验 8 求解常微分方程的初值问题

### 一、实验目的

用欧拉方法、改进的欧拉方法和四阶经典龙格-库塔方法求解常微分方程的初值问题, 并比较各种方法优缺点。

### 二、实验方法

- (1) 欧拉方法

$$y_{n+1} = y_n + hf(x_n, y_n)$$

- (2) 改进的欧拉方法

$$\begin{cases} T_1 = y_n + hf(x_n, y_n) \\ T_2 = y_n + hf(x_{n+1}, T_1) \\ y_{n+1} = \frac{T_1 + T_2}{2} \end{cases}$$

### (3) 四阶经典龙格-库塔方法

$$\begin{cases} y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 = f(x_n, y_n) \\ k_2 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1) \\ k_3 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2) \\ k_4 = f(x_n + h, y_n + hk_3) \end{cases}$$

## 三、实验内容

用欧拉方法、改进的欧拉方法和四阶经典龙格-库塔方法求解常微分方程的初值问题

$$\begin{cases} \frac{dy}{dx} = \frac{2}{3}x \cdot y^{-2} & x \in [0, 1] \\ y(0) = 1 \end{cases}$$

的数值解（取  $h = 0.1$ ），并将计算结果与准确解  $y = \sqrt[3]{1+x^2}$  进行比较。

## 四、实验程序

（略）

## 五、实验结果（仅供参考）

部分结果如下：

$x_i$	欧 拉 方 法	改进欧拉方法	四阶经典龙格-库塔方法	准 确 解
0.2	1.019 824	1.013 180	1.013 159	1.013 159 43
0.4	1.063 754	1.057 51	1.050 718	1.050 717 59
0.6	1.126 810	1.107 965	1.107 932	1.107 931 61
0.8	1.202 845	1.179 297	1.179 274	1.179 273 72
1.0	1.287 372	1.259 930	1.259 921	1.259 921 07

## 六、结果分析

（略）

提示：比较不同方法的计算量和优劣。

## 实验 9 求解三对角线性方程组

### 一、实验目的

用追赶法解三对角线方程组  $Ax = f$ ，并分析计算量。

$$A = \begin{pmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{pmatrix}, \quad f = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}$$

## 二、实验方法

将三对角方程  $Ax = f$  的系数矩阵分解成两个二对角矩阵的乘积。设  $A = LU$ ，且

$$A = \begin{pmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{pmatrix}, \quad L = \begin{pmatrix} l_1 & & & & \\ a_2 & l_2 & & & \\ & a_3 & l_3 & & \\ & & \ddots & \ddots & \\ & & & a_n & l_n \end{pmatrix}, \quad U = \begin{pmatrix} 1 & u_1 & & & \\ & 1 & u_2 & & \\ & & 1 & \ddots & \\ & & & \ddots & u_{n-1} \\ & & & & 1 \end{pmatrix}$$

这样，解方程组  $Ax = f$  就化为求  $LUx = f$ ，令  $Ux = y$ ，则  $Ly = f$ 。

解方程组  $Ly = f$ ，即

$$\begin{cases} l_1 y_1 = f_1 \\ a_i y_{i-1} + l_i y_i = f_i \end{cases} \quad (i = 2, \dots, n)$$

得

$$\begin{cases} y_1 = f_1 / l_1 \\ y_i = (f_i - a_i y_{i-1}) / l_i \end{cases} \quad (i = 2, \dots, n)$$

解方程组  $Ux = y$  即

$$\begin{cases} x_i + u_i x_{i+1} = y_i \\ x_n = y_n \end{cases} \quad (i = 1, 2, \dots, n)$$

得

$$\begin{cases} x_n = y_n \\ x_i = y_i - u_i x_{i+1} \end{cases} \quad (i = n-1, \dots, 2, 1)$$

## 三、实验内容

用追赶法解方程组

$$\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

## 四、实验程序

(略)

## 五、实验结果

$$y = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}\right), x = \left(\frac{5}{6}, \frac{2}{3}, \frac{1}{2}, \frac{1}{3}, \frac{1}{6}\right)$$

## 六、结果分析

(略)

提示：比较追赶法与高斯法的计算量和优劣。

## 实验 10 矩阵特征值问题计算

### 一、实验目的

用幂法、QR 方法、雅可比方法求矩阵的主特征值或全部特征值。

### 二、实验方法

#### 1. 幂法

输入初值  $v_0 = u_0 \neq 0$ ，以及精度和最大迭代次数  $\varepsilon, N$ 。

对于  $k=1, 2, \dots, N$  执行

$$\begin{aligned} v_k &= A u_{k-1} \\ \mu_k &= \max(v_k) \\ u_k &= v_k / \mu_k \end{aligned}$$

当  $|u_k - u_{k-1}| < \varepsilon$  时，结束，否则  $u_{k-1} \leftarrow u_k$ ，继续循环。

#### 2. QR 方法

令  $A_1 = A$ ，对  $A_1$  进行 QR 分解： $A_1 = Q_1 R_1$ ，然后令  $A_2 = R_1 Q_1$ ，再对  $A_2$  进行 QR 分解： $A_2 = Q_2 R_2$ ，然后令  $A_3 = R_2 Q_2$ ，这样继续下去，便得到一个矩阵序列  $\{A_k\}$ ，即

$$\begin{cases} A_1 = A \\ A_k = Q_k R_k \\ A_{k+1} = R_k Q_k \end{cases} \quad (k=1, 2, \dots)$$

矩阵序列  $\{A_k\}$  与  $A$  有相同的特征值。而矩阵序列  $\{A_k\}$  本质上收敛于上三角矩阵或块上三角矩阵，且对角块为  $1 \times 1$  或  $2 \times 2$  矩阵。 $1 \times 1$  矩阵就是  $A$  的实特征值。每个  $2 \times 2$  矩阵都含有  $A$  的一对复特征值。然后，可以用反幂法求对应的特征向量。

#### 3. 雅可比方法

输入矩阵  $A$ ，精度和最大迭代次数  $\varepsilon, N$ ，令  $R = I$ 。

对于  $k=1, 2, \dots, N$  执行以下步骤：

计算  $A = R A R^T$

找最大值  $m = a_{pq} = \max_{1 \leq i, j \leq n; i \neq j} |a_{ij}|$

计算  $\theta$ , 使  $\tan 2\theta = \frac{2a_{pq}}{a_{pp} - a_{qq}}$

利用式 (10.4.1) 计算  $R = R(p, q, \theta)$ , 若  $m < \varepsilon$ , 则结束。

### 三、实验内容

用幂法、QR 方法、雅可比方法求矩阵  $A$  的主特征值或全部特征值。

$$A = \begin{pmatrix} 4 & 2 & 2 \\ 2 & 5 & 1 \\ 2 & 1 & 6 \end{pmatrix}$$

### 四、实验程序

(略)

### 五、实验结果

$\lambda_1 = 0.3874$ ,  $x_1 = (0.8077, 0.7720, 1)^T$ ,  $\lambda_2 = 4.4867$ ,  $x_2 = (0.2170, 1, -0.9473)^T$

$\lambda_3 = 2.2160$ ,  $x_3 = (1, -0.5673, -36998)^T$

### 六、结果分析

(略)

提示: 比较不同方法的计算量和优劣。

## 实验 11 函数优化计算

### 一、实验目的

用梯度法、牛顿法、共轭方向法求多元函数的最优解。

### 二、实验方法

#### 1. 梯度法的算法步骤

- ① 给出初始点  $\mathbf{x}^0$  和计算精度  $\varepsilon > 0$ , 并令  $k=0$ 。
- ② 计算  $\nabla f(\mathbf{x}^k)$ 。
- ③ 若  $\|\nabla f(\mathbf{x}^k)\| < \varepsilon$ , 则迭代结束, 取  $\mathbf{x}^* = \mathbf{x}^k$ , 否则转④。
- ④ 进行一维搜索, 由  $\min_{t \geq 0} f(\mathbf{x}) = \min_{t \geq 0} f(\mathbf{x}^k - t \nabla f(\mathbf{x}^k))$  求得  $t$ , 记为  $t_k$ 。
- ⑤ 令  $\mathbf{x}^{k+1} = \mathbf{x}^k - t_k \nabla f(\mathbf{x}^k)$ ,  $k = k+1$ , 返回②。

## 2. 牛顿法的算法步骤

- ① 给出初始点  $\mathbf{x}^0$  和计算精度  $\varepsilon > 0$ ，并令  $k = 0$ 。
- ② 计算  $\nabla f(\mathbf{x}^k)$ ，如果  $\|\nabla f(\mathbf{x}^k)\| < \varepsilon$ ，则迭代结束，取  $\mathbf{x}^* = \mathbf{x}^k$ ，否则转③。
- ③ 计算  $[\nabla^2 f(\mathbf{x}_k)]^{-1}$  与  $[\nabla^2 f(\mathbf{x}^k)]^{-1} \nabla f(\mathbf{x}_k)$ 。
- ④ 令  $\mathbf{x}^{k+1} = \mathbf{x}^k - [\nabla^2 f(\mathbf{x}_k)]^{-1} \nabla f(\mathbf{x}_k)$ ， $k = k + 1$ ，返回②。

## 3. 共轭方向法的计算步骤

- ① 给出初始点  $\mathbf{x}^0$  和计算精度  $\varepsilon > 0$ 。
- ② 计算  $\mathbf{g}_0 = \nabla f(\mathbf{x}^0)$ ，令  $\mathbf{s}^0 = -\mathbf{g}_0, k = 0$ 。
- ③ 进行一维搜索  $\min_{t \geq 0} f(\mathbf{x}^k + t\mathbf{s}^k)$ ，求得  $t$ ，记为  $t_k$ ，计算

$$\mathbf{x}^{k+1} = \mathbf{x}^k + t\mathbf{s}^k, \quad \mathbf{g}_{k+1} = \nabla f(\mathbf{x}^{k+1})$$

- ④ 如果  $\|\mathbf{g}_{k+1}\| < \varepsilon$ ，则迭代结束，取  $\mathbf{x}^* = \mathbf{x}^{k+1}$ ，否则转⑤。
- ⑤ 如果  $k < n - 1$ ，利用式 (11.3.9) 计算  $\lambda_{k+1,k}$ ，即

$$\lambda_{k+1,k} = \frac{\|\mathbf{g}_{k+1}\|^2}{\|\mathbf{g}_k\|^2}, \quad \mathbf{s}^{k+1} = -\mathbf{g}_{k+1} + \lambda_{k+1,k} \mathbf{s}^k$$

令  $k = k + 1$ ，返回③。

如果  $k = n - 1$ ，令  $\mathbf{x}^0 = \mathbf{x}^n$ ，返回②。

## 三、实验内容

用梯度法、牛顿法、共轭方向法求多元函数的最优解。

式中  $f(\mathbf{x}) = (x_1^3 - x_2)^2 + 100(1 - x_1^2)^2$ 。

## 四、实验程序

(略)

## 五、实验结果

$$\mathbf{x}^* = (1, 1), f^* = 0$$

## 六、结果分析

(略)

提示：比较不同方法的计算量和优劣。

## 参 考 文 献

- [1] 李庆扬, 王能超, 易大义. 数值分析 (第 4 版). 北京: 清华大学出版社, 2001.
- [2] 李庆扬. 数值分析基础教程. 北京: 高等教育出版社, 2001.
- [3] 李庆扬. 数值分析复习与考试指导. 北京: 高等教育出版社, 2000.
- [4] 姜健飞, 胡良剑, 唐俭. 数值分析及其 MATLAB 实验. 北京: 科学出版社, 2004.
- [5] 王沫然. MATLAB 5.X 与计算方法. 北京: 清华大学出版社, 2000.
- [6] Shoichiro Nakamura. 科学计算引论——基于 MATLAB 的数值分析. 梁恒, 刘晓艳, 等译. 北京: 电子工业出版社, 2002.
- [7] 王能超. 数值分析简明教程 (第 2 版). 北京: 高等教育出版社, 2003.
- [8] 同济大学计算数学教研社. 现代数值数学和计算. 上海: 同济大学出版社, 2004.
- [9] 陈宝林. 最优化理论与算法. 北京: 清华大学出版社, 2005.
- [10] 金聪, 熊盛武. 数值分析. 武汉: 武汉理工大学出版社, 2003.
- [11] 魏毅强, 张建国, 张洪斌, 等. 数值计算方法. 北京: 科学出版社, 2004.
- [12] 高培旺. 计算方法典型例题与习题. 长沙: 国防科技大学出版社, 2003.
- [13] 封建湖, 聂玉峰, 王振海. 数值分析 (第 4 版) 导教·导学·导考. 西安: 西北工业大学出版社, 2003.
- [14] 陈延梅, 吴勃英, 金承日. 计算方法学习指导. 北京: 科学出版社, 2003.
- [15] 吴筑筑. 计算方法. 北京: 清华大学出版社, 2004.
- [16] 黄华江. 实用化工计算机模拟. 北京: 化学工业出版社, 2004.
- [17] 李乃成, 邓建中. 数值计算方法. 西安: 西安交通大学出版社, 2002.
- [18] 李庆扬. 科学计算方法. 北京: 清华大学出版社, 2006.
- [19] 吴勃英. 数值分析原理. 北京: 科学出版社, 2003.
- [20] 施浒立, 赵彦. 误差设计新理念与方法. 北京: 科学出版社, 2007.
- [21] 陈传淼. 科学计算概论. 北京: 科学出版社, 2007.
- [22] 袁亚湘. 非线性优化计算. 北京: 科学出版社, 2008.
- [23] 马良等. 蚁群优化计算. 北京: 科学出版社, 2008.
- [24] 阳明盛, 罗长童. 最优化原理、方法及求解软件. 北京: 科学出版社, 2006.
- [25] 孙文瑜. 最优化方法. 北京: 高教出版社, 2004.
- [26] 李庆扬, 王能超, 易大义. 数值分析 (第 5 版). 北京: 清华大学出版社, 2008.
- [27] 关治, 陆金甫. 数值方法. 北京: 清华大学出版社, 2006.
- [28] 马昌凤, 林伟川. 现代数值计算方法. 北京: 科学出版社, 2008.
- [29] 陈宝林. 最优化理论与算法. 北京: 清华大学出版社, 2005.
- [30] John H Mathews, Kurtis D Fink. Numerical Methods Using MATLAB(Fourth Edition). 北京: 电子工业出版社, 2007.





欢迎登录 免费 获取本书教学资源  
<http://www.hxedu.com.cn>



## 计算方法 (第2版)

本书比较全面地介绍了现代科学与工程计算中常用的数值计算方法。全书共分12章,主要内容有:引论、计算方法的数学基础、方程求根、解线性方程组的直接法、解线性方程组的迭代法、函数插值、函数逼近、数值积分与数值微分、常微分方程初值问题的数值解法、矩阵特征值计算、函数优化计算和MATLAB编程基础及其在计算方法中的应用。

本书知识体系完整,从简要回顾与计算方法有关的数学基础知识,到介绍现代计算软件MATLAB在本领域中的应用,书中每个算法都配有结构化流程图,大部分算法给出了MATLAB语言和C语言的源代码,书后附上上机实验题目。可从华信教育资源网([www.hxedu.com.cn](http://www.hxedu.com.cn))免费下载的教学资源包括:电子教案、各章习题解答和模拟试题。

本书可作为高等院校理工科计算机、电子信息类及近电类本科和研究生教材使用,也可供从事科学与工程计算的科技工作者和研究人员参考。



责任编辑:冉哲  
封面设计:徐海燕

ISBN 978-7-121-20328-2



9 787121 203282 >

定价:41.00元